CONFIDENTIAL — SAMPLE / DEMONSTRATION

iOS APPLICATION
PENETRATION TESTING REPORT

NovaSphere Pay — Fictional Sample Assessment

2026 Illustrative Sample Deliverable

Organization	NovaSphere Digital Technologies, Inc. (Fictional)
Application	NovaSphere Pay — com.novasphere.pay.ios (Fictional)
Prepared By	TMG Security
Assessment Type	iOS Application Security Assessment / Penetration Test
Assessment Period	17 August 2026 – 08 September 2026
Report Date	11 September 2026
Classification	CONFIDENTIAL — SAMPLE / DEMONSTRATION

FICTIONAL SAMPLE — NOT AN ACTUAL CLIENT iOS PENETRATION TEST

All findings, evidence, and results are illustrative.

DOCUMENT CONTROL

Document Title	iOS Application Penetration Testing Report — NovaSphere Pay (Fictional Sample)
Document ID	TMG-IOS-NSP-2026-001
Version	1.0
Issue Date	11 September 2026
Assessment Period	17 August 2026 – 08 September 2026
Assessment Type	iOS Application Security Assessment / Penetration Test
Client	NovaSphere Digital Technologies, Inc. (Fictional)
Application	NovaSphere Pay — com.novasphere.pay.ios (Fictional)
IPA Under Test	NovaSpherePay-6.4.2.ipa (Fictional, Build 6402)
Backend API	https://api.novasphere-example.com (Fictional, non-resolving)
Auth Service	https://auth.novasphere-example.com (Fictional, non-resolving)
Environment	Fictional staging / controlled assessment environment
Standards Referenced	OWASP MASVS (current), OWASP MASTG v2.0.0, OWASP MASWE v1.0.0
Prepared By	TMG Security Offensive Security Team
Technical Lead	TMG Security — Lead iOS Penetration Tester (Fictional Engagement Role)
Reviewed By	TMG Security Quality Assurance Lead
Classification	CONFIDENTIAL — SAMPLE / DEMONSTRATION

Version History

Version	Date	Author	Summary
1.0	11 September 2026	TMG Security Offensive Security Team	Initial issue — fictional sample deliverable.

Distribution List

The following fictional distribution list reflects the recipients TMG Security would designate for an engagement of this type; no real individuals are named.

Recipient (Fictional Role)	Organization	Purpose
Chief Information Security Officer	NovaSphere Digital Technologies, Inc.	Executive sponsor and primary report recipient.
VP of Mobile Engineering	NovaSphere Digital Technologies, Inc.	Remediation ownership and engineering prioritization.
Director of Product Security	NovaSphere Digital Technologies, Inc.	Coordinates retest scheduling and finding tracking.
Engagement Manager	TMG Security	Client relationship management and delivery oversight.
Quality Assurance Lead	TMG Security	Independent technical review of this report prior to issue.

Authorship

Name (Fictional Role)	Role	Contribution
Elena Marquez	Lead iOS Penetration Tester	Static and dynamic testing, finding authorship, report lead.
Marcus Whitfield	Mobile Application Security Consultant	Mobile-API integration testing, WKWebView and IPC testing.
Priya Raman	Quality Assurance Lead	Independent technical review and report quality assurance.

Name (Fictional Role)	Role	Contribution
David Okafor	Engagement Manager	Scoping, rules of engagement, and client coordination.

IMPORTANT DISCLAIMER

FICTIONAL SAMPLE REPORT — NOT AN ACTUAL CLIENT iOS PENETRATION TEST

This report has been produced solely for demonstration and portfolio purposes on behalf of TMG Security. It is intended to illustrate the structure, technical depth, and professional reporting standard of an iOS application penetration testing deliverable built on current OWASP Mobile Application Security (MASVS / MASTG / MASWE) material, and is presented as the iOS counterpart to TMG Security's companion Android sample report. To ensure there is no ambiguity regarding the nature of this document, TMG Security sets out the following clarifications:

- NovaSphere Digital Technologies, Inc. is a fictional entity. It does not correspond to any real company, and any resemblance to an actual organization is purely coincidental.
- NovaSphere Pay (bundle identifier com.novasphere.pay.ios) is a fictional iOS application. No real production application was built, signed, distributed, or accessed.
- The IPA file referenced throughout this report (NovaSpherePay-6.4.2.ipa) is a fictional build artifact; no real binary of this name exists.
- The backend domains referenced throughout this report (api.novasphere-example.com, auth.novasphere-example.com) are fictional, non-resolving placeholder domains used only for illustration.
- No real client was engaged and no real client iOS application, backend infrastructure, or mobile device fleet was tested.
- No production environment, system, network, cloud resource, or App Store listing of any kind was accessed, scanned, or tested in the creation of this report.
- No real user data, personal information, payment card data, or customer records were accessed, viewed, or processed.
- No real credentials — including device, application, API, signing, provisioning-profile, or third-party SDK credentials — were used or are disclosed anywhere in this report.
- All Info.plist excerpts, entitlements excerpts, Mach-O binary/otool/strings/nm output, Swift and Objective-C code fragments, Keychain dumps, and network traffic captures shown in this report are synthetic and constructed for illustration only.
- All vulnerabilities, findings, risk ratings, CVSS-style scores, dates, and remediation statuses described in this report are fictional and were constructed for demonstration purposes.
- All Proofs of Concept (PoCs) are illustrative, synthetic, and non-destructive; none were executed against a real or live target, real device population, or production backend.
- All screenshots and tool-style evidence panels (Xcode, otool/strings-style output, decompiler-style output, MASVS-style dashboards, interception-proxy-style traffic, Console.app/log-style output) are synthetic mock-ups created specifically for this sample report, not real tool captures.
- No real exploitation occurred at any point during the preparation of this document, and no actual customer impact occurred.
- TMG Security is not affiliated with, endorsed by, reviewed by, or certified by OWASP; references to OWASP MASVS, MASTG, and MASWE describe the testing framework used and do not imply any official certification of NovaSphere Pay or of TMG Security.
- This report does not represent an actual TMG Security client engagement and must not be relied upon as evidence of the security posture of any real organization or application.

Intended Use

This sample report may be shared with prospective clients, partners, and other interested parties to illustrate TMG Security's iOS application security testing methodology, technical reporting standards, and evidence-presentation quality. It should not be relied upon for any compliance, legal, contractual, or procurement decision, and does not constitute security advice regarding any real application.

TABLE OF CONTENTS

01 EXECUTIVE SUMMARY

TMG Security has prepared this fictional, illustrative iOS Application Penetration Testing report to represent NovaSphere Digital Technologies, Inc., a hypothetical U.S.-based FinTech / digital payments organization headquartered in Austin, Texas, and its fictional flagship consumer application, NovaSphere Pay (bundle identifier com.novasphere.pay.ios). This section presents a fictional executive-level summary of the simulated assessment — covering IPA static analysis, dynamic on-device analysis, and mobile-API integration testing against current OWASP MASVS, MASTG v2.0.0, and MASWE v1.0.0 material — prepared as the iOS counterpart to TMG Security's companion Android sample report, in the style and format TMG Security would use for an actual client engagement.

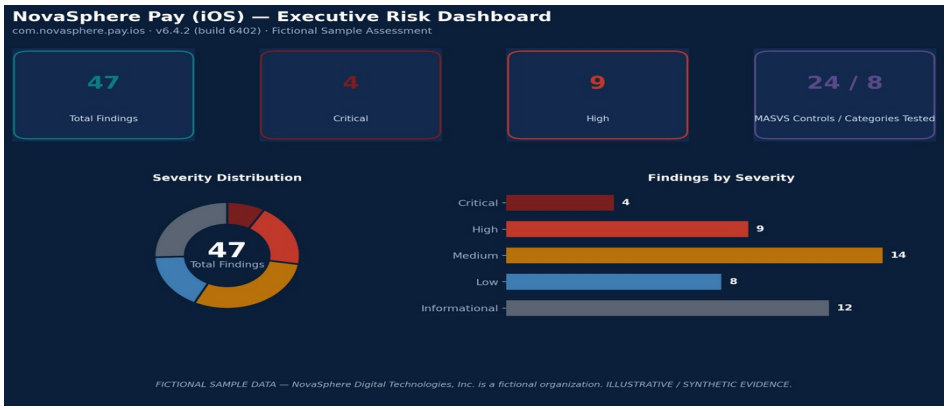
ALL METRICS, FINDINGS, AND SCORES ON THIS PAGE ARE ILLUSTRATIVE SAMPLE DATA

Overall Illustrative Security Posture

REQUIRES REMEDIATION

NovaSphere Pay's simulated iOS build demonstrates a number of established security controls — including consistent multi-factor enforcement on account recovery, strong TLS configuration on its primary API endpoints, and a properly signed, non-debuggable release build — but the fictional assessment identified material weaknesses concentrated in local/Keychain data protection, mobile-API object-level authorization, and the trust boundary around Universal Links, custom URL schemes, and WKWebView bridges. These require prioritized remediation before the application should be considered ready for a broader trust posture. This illustrative status reflects a sample iOS penetration-testing exercise and does not represent a certification or compliance decision, and does not imply any official OWASP certification.

Fictional Sample Dashboard



Illustrative severity distribution across all 47 fictional findings — ILLUSTRATIVE SAMPLE DATA.

Executive Risk Narrative

The fictional assessment identified 47 illustrative findings across NovaSphere Pay's iOS client, on-device attack surface, and mobile-API integration: 4 Critical, 9 High, 14 Medium, 8 Low, and 12 Informational. The four Critical findings concern the most fundamental trust-boundary failures observed: session and refresh tokens persisted in a plaintext NSUserDefaults plist rather than the Keychain (IOS-001), a Keychain item gating biometric re-authentication stored with the least restrictive accessibility class and no access-control policy (IOS-002), broken object-level authorization on the mobile-only funds-transfer confirmation endpoint allowing one authenticated user to move funds out of another user's account (IOS-003), and a Universal Link flow that silently commits a new payee association the instant the link is opened, with no in-app confirmation (IOS-004).

The nine High findings span an unreserved custom URL scheme that can be claimed by any co-installed app and used to hijack an OAuth account-linking callback (IOS-005), a WKWebView JavaScript bridge that returns the live session token to any calling script with no origin verification (IOS-006), a certificate-validation defect in a legacy statement-export network client that accepts a substituted certificate under a specific interception scenario (IOS-007), verbose os_log output in a QA build configuration that writes the live bearer token and MFA code to the unified log (IOS-008), a hardcoded third-party mapping-SDK key recoverable via a plain strings extraction of the Mach-O binary (IOS-009), an App Group container shared with the Home Screen widget that stores a cached balance and partial card number in an unencrypted plist (IOS-010), a Keychain access-group misconfiguration (IOS-011), a biometric app-unlock gate that is not properly bound to a LocalAuthentication/Keychain operation and can be bypassed on a compromised device (IOS-013), and a token-lifecycle gap in which logout does not trigger server-side refresh-token revocation (IOS-014). Consistent with the pattern observed in the Critical findings, these High-severity issues concentrate around local/Keychain data protection and the IPC/URL-scheme/WebView trust boundary — precisely the areas OWASP MASVS-STORAGE and MASVS-PLATFORM are designed to address.

Medium-severity findings reflect defense-in-depth and hardening gaps — sensitive balances readable via the pasteboard and exposed in the background-app-snapshot thumbnail, an overly permissive document-interaction/file-sharing configuration, incomplete data isolation for an App Extension, jailbreak/runtime-integrity controls relying on bypassable client-side heuristics, sensitive data reaching push-notification payloads, a client-side-only session timeout, an ATS exception permitting weak TLS on a secondary endpoint, broader-than-necessary privacy permission requests, an analytics SDK that does not honor identifier resets across reinstall, insecure client-side random-number generation for a security-relevant value, verbose API error responses revealing internal implementation detail, insecure object persistence of a sensitive local cache, and missing per-domain granularity in the App Transport Security configuration. Low and Informational findings are hardening and governance opportunities — unused permission declarations, deprecated API usage, logging hygiene in non-sensitive modules, third-party dependency governance, missing automated security-regression coverage, and positive-control observations such as consistent multi-factor enforcement on account recovery and strong TLS on primary API endpoints — that support a stronger long-term security posture without representing an immediate exploitation path in this fictional sample.

TMG Security's simulated recommendation is that NovaSphere prioritize closure of the four Critical and nine High findings within the first 60 days of the illustrative remediation roadmap (see Section 76), with particular attention to Keychain-backed local storage, object-level authorization on mobile-API endpoints, and mandatory in-app confirmation for any Universal Link, custom-URL-scheme, or deep-link flow that changes account state. Medium findings should be addressed

within 90 days, and Low/Informational items folded into the ongoing defense-in-depth and mobile-governance backlog. A full retest is recommended following remediation of all Critical and High findings, per the retest methodology in Section 77.

02 ENGAGEMENT OVERVIEW

This section describes the fictional objective, type, and approach of the illustrative NovaSphere Pay iOS application penetration test, prepared to demonstrate TMG Security's engagement-scoping and reporting standards for mobile application security assessments, as the iOS counterpart to TMG Security's companion Android sample report.

Assessment Objective

The fictional objective of this engagement was to identify and validate exploitable security weaknesses across NovaSphere Pay's iOS client — spanning local/Keychain data storage, cryptography, network communication, authentication and biometric local authentication, IPC surfaces (custom URL schemes, Universal Links, deep links, App Extensions, App Groups), WKWebView usage, resilience against reverse engineering and tampering, privacy controls, and the application's integration with its backend mobile API — and to provide NovaSphere's engineering leadership with prioritized, actionable remediation guidance aligned to current OWASP Mobile Application Security material.

Assessment Type & Testing Window

This is a fictional iOS Application Security Assessment / Penetration Test conducted against a controlled, fictional staging environment and a fictional release-candidate build (NovaSpherePay-6.4.2.ipa, build 6402) over the illustrative assessment period of 17 August 2026 – 08 September 2026, with this report issued 11 September 2026.

Testing Approach

- Static analysis of the fictional IPA: Info.plist and entitlements review, provisioning-profile and code-signing validation, Mach-O binary and Swift/Objective-C static analysis, hardcoded-secret discovery, and third-party SDK/dependency inventory.
- Dynamic analysis on fictional jailbroken and non-jailbroken physical reference devices: runtime Keychain and local-storage inspection, network-traffic interception, authentication and biometric flow testing, custom-URL-scheme, Universal Link, and deep-link testing.
- Mobile-API integration testing of the backend endpoints NovaSphere Pay calls, assessed from the perspective of the iOS client's authorization and data-handling behavior.
- Structured mapping of every test performed to the current OWASP MASVS control set, using OWASP MASTG v2.0.0 iOS-specific test concepts and OWASP MASWE v1.0.0 weakness concepts wherever a verifiable identifier applies.
- Resilience testing of anti-tampering, anti-reverse-engineering, jailbreak/debugger-detection, and binary-protection controls using standard, non-destructive mobile security testing techniques.
- Privacy-focused review of permission usage, PII handling, and third-party analytics/tracking SDK behavior.

This engagement overview describes a fictional, illustrative assessment. No real client engagement occurred.

03 SCOPE

The following fictional scope table defines the boundaries of this illustrative assessment.

In Scope

Area	Detail
iOS Application (IPA)	NovaSpherePay-6.4.2.ipa (com.novasphere.pay.ios), build 6402, minimum iOS 16.0, tested against iOS 18.x (fictional)
Static Analysis	Info.plist, entitlements, provisioning profile, code signing, Mach-O binary, Swift/Objective-C static review, third-party SDK inventory

Area	Detail
Dynamic Analysis	Runtime Keychain/local storage, network traffic, authentication flows, custom URL schemes, Universal Links, deep links
Local Data Storage	Keychain, NSUserDefaults/UserDefaults, App Group shared containers, on-disk cache, pasteboard, backups
Cryptography	Key management, CryptoKit/Security framework usage, algorithm/mode selection, randomness
Network Communication	App Transport Security configuration, TLS validation, certificate/public-key pinning, custom URLSessionDelegate trust logic
Authentication & Authorization	Login, MFA, LocalAuthentication (Face ID/Touch ID), session/token handling, token lifecycle
Mobile-API Integration	Authorization and data-handling behavior of backend endpoints called by the iOS client, reachable from the app's own network traffic
IPC / Attack Surface	Custom URL schemes, Universal Links, deep links, App Extensions, App Groups, Handoff, SiriKit/App Intents
WKWebView	JavaScript bridge exposure, navigation policy, content-source restrictions
Resilience	Jailbreak detection, anti-debugging, anti-tampering, binary security protections, reverse-engineering resistance
Privacy	Privacy permission usage (App Tracking Transparency and Info.plist usage-description keys), PII handling, third-party analytics/tracking SDK behavior
Universal Link Domain	app.novasphere-example.com (fictional, associated-domains/apple-app-site-association configuration)
App Extensions In Scope	App Extensions — Home Screen Widget (WidgetKit), Share/Action Extension, Notification Service Extension, and Notification Content Extension

Out of Scope

Area	Detail
Android Application	Not included; this engagement is iOS-only (see the companion Android sample report)
Denial-of-Service Testing	Not performed under any circumstances
Destructive Testing	Not performed under any circumstances
Social Engineering	Not included in this engagement
Physical Security	Not included in this engagement
Production Infrastructure	Testing restricted to the fictional staging environment and fictional test devices only
Backend Infrastructure Internals	Server-side infrastructure hardening beyond mobile-API authorization behavior not tested
App Store Review Process	Apple's own App Store review, notarization, and submission process was not tested or evaluated
Signing-Certificate / Provisioning-Profile Custody	Not directly tested in this engagement — see IOS-047 and Section 68
Third-Party Backend Systems	Not tested; only NovaSphere-owned fictional application and API surface in scope

04 OUT-OF-SCOPE

Beyond the area-level scope boundaries in Section 3, the following specific fictional components and considerations were explicitly excluded from direct testing in this engagement, and are flagged here for transparency.

Component / Area	Reason Excluded
Backend infrastructure penetration test	Separate assessment discipline; this engagement is limited to mobile-API authorization and data-handling behavior reachable from the iOS client
App Store review process	Apple's own submission, notarization, and review pipeline is outside TMG Security's assessment scope
Physical and social engineering	Not included in this engagement's rules of engagement

Component / Area	Reason Excluded
Mapping/geolocation SDK vendor infrastructure	Third-party managed service; only client-side integration reviewed (see IOS-009)
Analytics SDK vendor infrastructure	Third-party managed service; only client-side configuration reviewed (see IOS-023, Appendix M)
Crash-reporting SDK vendor infrastructure	Third-party managed service; only client-side configuration reviewed (see IOS-043, Appendix M)
Bank-account aggregation partner infrastructure	Third-party managed OAuth partner service; only client-side callback handling reviewed (see IOS-005)
Physical device supply chain / OEM firmware	Outside the fictional engagement's mobile-application-security scope
Push-notification service provider infrastructure	Third-party managed service (Apple Push Notification service); only client-side handling reviewed (see Section 50)

Exclusion from direct testing does not imply these components are free of risk; it reflects this fictional engagement's defined scope and duration.

05 APPLICATION PROFILE

NovaSphere Pay is a fictional consumer FinTech / digital payments iOS application created to demonstrate a realistic mobile application assessment scope. All organization, technology, and infrastructure detail below is illustrative.

Fictional Client Profile

Attribute	Detail
Organization	NovaSphere Digital Technologies, Inc. (Fictional)
Industry	FinTech / Digital Payments / Consumer Technology
Headquarters	Austin, Texas, USA (Fictional)
Employees	Approximately 850 (Fictional)
Application	NovaSphere Pay (Fictional)
Bundle Identifier	com.novasphere.pay.ios
Distribution	Apple App Store (fictional listing; illustrative build only)
Backend API	https://api.novasphere-example.com (Fictional, non-resolving placeholder)
Auth Service	https://auth.novasphere-example.com (Fictional, non-resolving placeholder)
Environment	Fictional staging / controlled assessment environment

Application Build Under Test

Attribute	Detail
IPA File Name	NovaSpherePay-6.4.2.ipa
Bundle Identifier	com.novasphere.pay.ios
Version / Build	6.4.2 (Build 6402)
Minimum iOS Deployment Target	16.0
Tested iOS Versions	18.x (fictional reference devices)
Architecture	arm64
Primary Language / Runtime	Swift, with a small number of legacy Objective-C bridging modules
Key Frameworks	UIKit, SwiftUI, Foundation, Security, LocalAuthentication, WebKit, CryptoKit, UserNotifications, AuthenticationServices
Signing	Apple Distribution certificate + App Store provisioning profile (fictional production identity)

Core Application Functionality (Fictional, Illustrative)

- Account login, MFA, and biometric app-unlock via LocalAuthentication (Face ID / Touch ID).
- Account balance, statement, and transaction-history viewing.
- Peer-to-peer and bill-pay funds transfer, including payee management via in-app flows and Universal Links.
- Card linking and card-detail viewing (masked in the UI).
- Branch-locator (map-based) and in-app help/support content, including a support WKWebView.
- Push-notification-based transaction and security alerts via UserNotifications.
- A Home Screen widget extension (SwiftUI) displaying an at-a-glance account balance via a shared App Group container.
- Third-party bank-account aggregation account linking via an OAuth flow presented through AuthenticationServices.

All infrastructure and technology detail above is fictional and provided only to give this sample report realistic structure and depth.

06 TESTING METHODOLOGY

TMG Security's iOS application penetration-testing methodology follows a structured, multi-phase process spanning static analysis, dynamic analysis, and mobile-API integration testing, built on current OWASP Mobile Application Security material — the OWASP Mobile Application Security Verification Standard (MASVS), the OWASP Mobile Application Security Testing Guide (MASTG) v2.0.0, and the OWASP Mobile Application Security Weakness Enumeration (MASWE) v1.0.0. No copyrighted material from these references is reproduced in this report; they are cited only as the conceptual and terminological framework for TMG Security's own methodology. TMG Security is not affiliated with, endorsed by, reviewed by, or certified by OWASP.

Methodology Phases

#	Phase	Illustrative Description
1	Reconnaissance & IPA Acquisition	Fictional IPA acquisition, integrity validation, and metadata extraction (Section 13).
2	Static Application Analysis	IPA structure, Info.plist, entitlements, provisioning profile, code signing, Mach-O binary, and Swift/Objective-C review (Sections 14-22).
3	Local Storage & Keychain Testing	Local storage, Keychain, and Keychain access-group testing (Sections 23-25).
4	Cryptography Testing	Key management, algorithm selection, and randomness review (Section 26).
5	Authentication & Authorization Testing	Login, MFA, biometric/local authentication, session, and token-lifecycle testing (Sections 27-30).
6	Network Security Testing	App Transport Security, TLS, and certificate-pinning testing (Sections 31-34).
7	Mobile-API Integration Testing	API communication and object-level authorization testing of backend endpoints called by the client (Sections 35-36).
8	WebView & IPC Attack Surface Testing	WKWebView, JavaScript bridge, custom URL scheme, Universal Link, deep link, App Extension, and App Group testing (Sections 37-49).
9	Platform Exposure Testing	Push notification, background snapshot, logging, and privacy-permission testing (Sections 50-53).
10	Resilience Testing	Reverse-engineering resistance, anti-tampering, jailbreak/debugger detection (Sections 56-60).
11	Vulnerability Validation	Manual confirmation of every candidate finding before inclusion in this report.
12	Reporting & Retest Planning	Structured finding documentation and defined retest scope (Section 77).

Static and Dynamic Analysis Toolchain (Illustrative)

Testing was performed using a fictional representative iOS security-testing toolchain, referenced throughout this report's synthetic evidence panels: IPA extraction and Mach-O static-analysis tooling (fictional static analysis tooling), a fictional interception proxy for dynamic traffic analysis, a fictional dynamic instrumentation framework for runtime hooking on jailbroken reference devices, a paired macOS workstation for Xcode-based build inspection and Console.app/log-style capture, and rooted-equivalent (jailbroken) and non-jailbroken fictional physical reference devices (Section 11). All tool

output shown in this report is synthetic and illustrative, not a real tool capture, and no specific commercial tool is represented as having been used against a real target.

07 RISK RATING METHODOLOGY

TMG Security assigns each finding a severity rating using language inspired by the CVSS v3.1 scoring framework, considering exploitability, privileges required, user interaction, scope, and technical/business impact. All scores and vectors in this report are illustrative sample estimations and were assigned for this fictional exercise only — they are not derived from any real vulnerability and are not submitted to any CVSS authority. Scores are presented in this report in the format used throughout, for example "9.1 (Critical)".

Severity Definitions

Severity	Illustrative Score Range	Definition
CRITICAL	9.0 – 10.0	Directly exploitable with severe confidentiality, integrity, or financial-transaction impact; typically requires immediate remediation.
HIGH	7.0 – 8.9	Significant impact with limited prerequisites; requires prompt remediation.
MEDIUM	4.0 – 6.9	Moderate impact, often requiring specific conditions (e.g., a jailbroken device) or providing defense-in-depth value when fixed.
LOW	0.1 – 3.9	Minor impact; hardening opportunity rather than a directly exploitable weakness.
INFORMATIONAL	N/A	Observation, positive-control note, or recommendation with no direct security-control failure.

Rating Factors Considered

- Exploitability — how readily the fictional weakness could be exercised on-device, over the network, or via the mobile-API surface.
- Privileges Required — the level of fictional device access needed before the weakness is reachable (e.g., standard use vs. jailbreak-level access).
- User Interaction — whether a fictional victim action (e.g., tapping a Universal Link) is required.
- Attack Complexity — conditions beyond the attacker's control that must align (e.g., network position, device state).
- Scope — whether the weakness affects only the vulnerable component or crosses a trust boundary (e.g., a local-storage gap that also enables mobile-API token replay).
- Technical Impact — effect on confidentiality, integrity, and availability of application and account data.
- Business Impact — fictional effect on fraud exposure, customer trust, and regulatory posture for a FinTech application.

IMPORTANT: All CVSS-style scores, vectors, severities, and risk ratings in this report are fictional and illustrative.

08 OWASP MASVS / MASTG / MASWE COVERAGE MODEL

This assessment is structured around the current OWASP Mobile Application Security (MAS) project material: the OWASP Mobile Application Security Verification Standard (MASVS), the OWASP Mobile Application Security Testing Guide (MASTG) v2.0.0, and the OWASP Mobile Application Security Weakness Enumeration (MASWE) v1.0.0. TMG Security uses these three references together as a single traceability chain, and applies a strict rule throughout this report: every MASVS, MASWE, or MASTG identifier cited is a real, verifiable identifier from current official OWASP material — no identifier in this report was invented. Where a finding's concept did not map to a precisely verifiable MASWE or MASTG identifier, this report says so explicitly rather than fabricating one.

The Traceability Chain

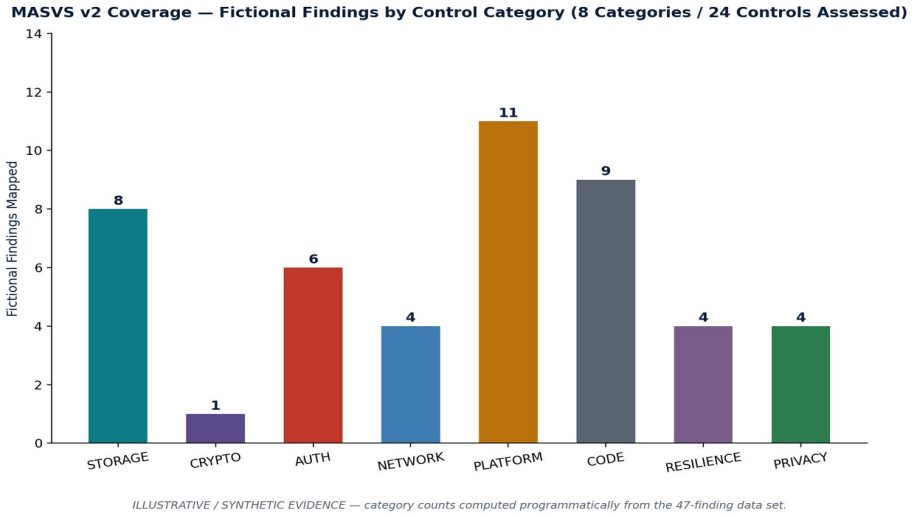
Each testing area in this report is traceable through four linked layers:

- MASVS Control — the security requirement being verified (e.g., MASVS-STORAGE-1: "The app securely stores sensitive data.").
- MASWE Weakness — the specific weakness pattern that would cause the control to fail (e.g., MASWE-0001: "Sensitive Data Stored Unencrypted in Private Storage.").
- MASTG Test — the current MASTG v2.0.0 iOS-specific test technique used to verify the control (e.g., MASTG-TEST-0300: "References to APIs for Storing Unencrypted Data in Private Storage.").
- Finding / Evidence — the fictional finding (if any) raised as a result, with its evidence reference in Appendix K.

MASVS and MASWE identifiers are platform-agnostic and share the same control/weakness text used in TMG Security's companion Android sample report. MASTG-TEST identifiers, however, are platform-specific: this report cites only the iOS-specific MASTG-TEST identifiers published under the MASTG's iOS test catalog, which use a different numeric range than the Android-specific MASTG-TEST identifiers cited in the companion Android report — the two are never interchanged in this document.

MASVS Control Groups Assessed

This assessment evaluated NovaSphere Pay against all eight current MASVS control groups, spanning 24 individual controls. Full traceability appears in Appendix H (MASVS Coverage Matrix) and Appendix I (MASWE/MASTG Traceability); the illustrative distribution of fictional findings across these eight groups is shown below.



Illustrative distribution of fictional findings across the 8 MASVS control groups — ILLUSTRATIVE SAMPLE DATA.

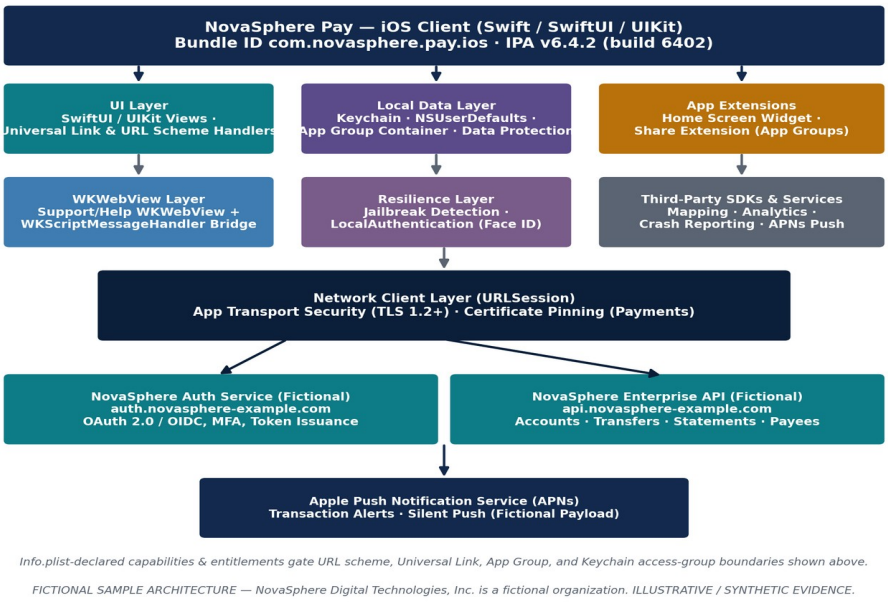
MASVS Group	Focus Area
MASVS-STORAGE	Secure local/Keychain data storage and prevention of sensitive-data leakage.
MASVS-CRYPTO	Strong cryptography and sound key management.
MASVS-AUTH	Secure authentication/authorization protocols, local authentication, and step-up authentication for sensitive operations.

MASVS Group	Focus Area
MASVS-NETWORK	Secure network traffic and identity pinning for developer-controlled endpoints.
MASVS-PLATFORM	Secure IPC (URL schemes, Universal Links, App Extensions), WebView usage, and user-interface handling.
MASVS-CODE	Up-to-date platform targeting, update enforcement, vulnerability-free components, input validation.
MASVS-RESILIENCE	Platform-integrity validation and resistance to tampering, static analysis, and dynamic analysis.
MASVS-PRIVACY	Minimized data/resource access, user non-identifiability, transparency, and user data control.

TMG Security is not affiliated with, endorsed by, reviewed by, or certified by OWASP. This assessment uses OWASP MASVS/MASTG/MASWE as a testing framework only; it does not constitute or imply official OWASP certification of NovaSphere Pay.

09 ASSESSMENT ARCHITECTURE

The following fictional architecture diagram illustrates how NovaSphere Pay's iOS client layers — the UIKit/SwiftUI presentation layer, local/Keychain data storage, the IPC surface (custom URL schemes, Universal Links, App Extensions, App Groups), the WKWebView layer, resilience controls, and third-party SDKs — sit atop a shared URLSession-based network client layer and communicate with the fictional NovaSphere Auth Service and Enterprise API. It is provided to give this sample report realistic technical grounding for the findings in later sections.



Illustrative fictional iOS application architecture — all service names and infrastructure detail are fictional.

Architecture Notes

- The UI Layer (UIKit and SwiftUI) hosts login, MFA, account, transfer, and Universal-Link/deep-link-handling screens; several of this report's Critical and High findings (IOS-003, IOS-004, IOS-005) trace to gaps at this layer's boundary with the network and IPC surfaces.
- The Local Data Layer spans the Keychain, UserDefaults/NSUserDefaults, the shared App Group container, and on-disk cache — the layer most directly implicated in IOS-001, IOS-002, IOS-010, and IOS-011.

- The IPC Surface (custom URL schemes, Universal Links, deep links, App Extensions, App Groups, Handoff, SiriKit/App Intents) is a recurring theme across this report's Critical findings, reflecting the outsized risk link-based and extension-based entry points carry in a FinTech application.
- The WKWebView Layer hosts the in-app support/help experience and its native JavaScript bridge, the subject of IOS-006.
- The Resilience Layer (jailbreak/debugger detection, binary security protections) is assessed in Sections 56-60 and is the subject of IOS-018 and IOS-039.
- All client layers converge on a single Network Client Layer (URLSession-based) enforcing App Transport Security, TLS validation, and certificate pinning on the primary payments channel, before reaching the fictional NovaSphere Auth Service and NovaSphere Enterprise API.
- Third-Party SDKs (mapping, analytics, crash reporting, bank-account aggregation partner) sit alongside the application's own modules and are reviewed in Section 54 and Appendix M.

10 IOS TEST ENVIRONMENT

This section describes the fictional test environment TMG Security constructed for this illustrative engagement, spanning device configuration, network interception, and instrumentation tooling. As with every other technical detail in this report, the environment described below is fictional and constructed for demonstration purposes.

Device Configuration

Testing combined fictional jailbroken and non-jailbroken physical reference devices spanning the application's supported iOS range (minimum iOS 16.0 through the tested iOS 18.x baseline), described fully in Section 11. Jailbroken fictional devices enabled filesystem-level inspection of the app container, Keychain enumeration, and runtime instrumentation; non-jailbroken fictional devices established a baseline for standard-condition control validation and confirmed that findings requiring jailbreak-level access were correctly scoped as such.

Network Interception

Dynamic network testing used a fictional interception proxy positioned on an isolated test network, configured with a fictional trusted root certificate installed on test devices to observe and, where relevant to a specific finding's reproduction steps, manipulate TLS traffic between the app and its fictional backend endpoints. All interception activity was confined to the dedicated fictional test devices and fictional test accounts described in Sections 11 and 12; no production traffic of any kind was observed or intercepted.

Instrumentation Toolchain

Runtime testing on fictional jailbroken devices used a fictional dynamic instrumentation framework to hook selected native and Swift runtime methods for tasks such as bypassing a client-side-only jailbreak-detection heuristic or observing in-memory token handling, strictly for the purpose of validating a specific fictional finding. A paired fictional macOS workstation running Xcode-equivalent tooling supported IPA extraction and validation, Mach-O binary inspection, and Console.app/log-style unified-log capture. No tool name in this section refers to a real product actually used against a real target; all toolchain references are generic and illustrative.

All device, network, and instrumentation detail in this section is fictional and describes an illustrative test environment only.

11 TEST DEVICES

The following fictional physical reference devices were used throughout this illustrative engagement, combining jailbroken and non-jailbroken configurations to cover both compromised-device and standard-condition testing scenarios, together with a fictional interception proxy and a paired macOS workstation used for static analysis and log capture. Full device detail is repeated in Appendix L for cross-reference.

Device	Configuration	Primary Use in This Engagement
Fictional Test Device A	Jailbroken iPhone 13 mini, iOS 16.4 (fictional checkra1n-style jailbreak)	Keychain dump, filesystem extraction, App Group container review, and jailbreak-detection bypass testing.
Fictional Test Device B	Non-Jailbroken iPhone 15 Pro, iOS 18.1	Baseline UX and control-validation testing under standard conditions.
Fictional Test Device C	Non-Jailbroken iPhone SE (3rd generation), iOS 16.0 (minimum-OS boundary)	Minimum-deployment-target behavior and backward-compatibility testing.
Fictional Test Device D	Jailbroken iPhone 12, iOS 17.5 (fictional jailbreak-hiding tweak installed)	Runtime instrumentation (Frida-style hooking), biometric-bypass, and jailbreak-hiding-tweak evasion testing.
Fictional Proxy Host	Interception Proxy (Burp-style) on Isolated Test Network	Network traffic interception, ATS/TLS, and certificate-pinning-resistance testing.
Fictional macOS Test Host	Paired macOS workstation (Xcode, Console.app, log CLI, libimobiledevice-equivalent tooling)	IPA extraction, otool/strings binary analysis, unified-log capture, and unencrypted local backup extraction.

All devices listed above are fictional test devices. No real device, real jailbreak, or real device fleet was used in the preparation of this report.

12 TEST ACCOUNTS

The following fictional, dedicated test accounts were provisioned for this illustrative engagement. No real user accounts, real customer data, or real credentials were used at any point in this fictional assessment; all credential values shown are placeholders. Full account detail is repeated in Appendix L for cross-reference.

Account	Role	Account ID	Credential	Purpose
ios.user01@example.test	Standard User (Test Account A)	ACCT-559002	[Fictional Test Credential]	Primary standard-user account — core flow testing and IOS-003 source-account role.
ios.user02@example.test	Standard User (Test Account B)	ACCT-771123	[Fictional Test Credential]	Secondary account — cross-account / object-level-authorization isolation testing (IOS-003).
ios.newuser01@example.test	Standard User (New Enrollment)	ACCT-559099	[Fictional Test Credential]	Onboarding, MFA enrollment, and biometric-registration flow testing.
ios.qa-admin01@example.test	QA / Internal Test Account	ACCT-000411	[Fictional Test Credential]	Used for QA-build verbose-logging capture (IOS-008) and pre-production configuration review.
ios.attacker-sim01@example.test	Standard User (Simulated Low-Privilege)	ACCT-990001	[Fictional Test Credential]	Simulated-attacker account for authorization, Universal Link, and custom URL scheme testing.

All test accounts listed above are fictional and were provisioned exclusively for this illustrative engagement. No real customer accounts were accessed or used.

13 IPA ACQUISITION & VALIDATION

Before static and dynamic testing began, TMG Security acquired and validated the fictional release-candidate build under test, confirming its integrity and provenance consistent with the methodology a real engagement would follow.

Fictional Acquisition Method

The fictional IPA (NovaSpherePay-6.4.2.ipa, build 6402) was provided directly by NovaSphere's fictional mobile engineering team via a fictional enterprise/TestFlight-equivalent distribution channel used for pre-release validation builds, consistent with how TMG Security typically receives a client-provided release-candidate build rather than sourcing it from the public App Store. The build was transferred to TMG Security's fictional test environment over an authenticated, access-controlled channel and installed on all fictional test devices listed in Section 11.

Illustrative Validation Tests Performed

- Verify the IPA's SHA-256 checksum matches the fictional value provided by NovaSphere's release pipeline, confirming no unauthorized modification occurred in transit.
- Verify the bundle identifier (com.novasphere.pay.ios) and bundle version/build (6.4.2 / 6402) match the fictional release intended for testing.
- Verify the code-signing signature is present and validates against the fictional Apple Distribution certificate and App Store provisioning profile (see Section 18).
- Verify the IPA was installable and launched correctly on all fictional test devices, including the minimum-deployment-target reference device (iOS 16.0).

Illustrative Observations

The fictional IPA was confirmed to match its expected checksum and version/build metadata, installed and launched correctly on all fictional test devices, and authenticated successfully using the fictional test accounts in Section 12 before testing began. No integrity or provenance issues were identified during acquisition.

Fictional IPA Summary

Attribute	Value
IPA File Name	NovaSpherePay-6.4.2.ipa
Bundle Identifier	com.novasphere.pay.ios
Bundle Short Version String / Bundle Version	6.4.2 (build 6402)
Minimum iOS Deployment Target	16.0
Target / Compiled SDK	iOS 18 SDK
Supported Architecture	arm64 (arm64e not observed)
Primary Language / Runtime	Swift 5.x, with a small number of legacy Objective-C bridging headers
Code Signing Scheme	Apple Distribution certificate + App Store provisioning profile (fictional production identity; no real fingerprint disclosed)

Full IPA metadata detail, including the complete third-party SDK inventory and embedded native library list, appears in Appendix B.

14 IPA STRUCTURE ANALYSIS

TMG Security unzipped the fictional release IPA (NovaSpherePay-6.4.2.ipa) and reviewed its on-disk package structure — the Payload/ directory, the main application bundle, embedded third-party frameworks, App Extension PlugIns, the embedded provisioning profile and entitlements, and the code signature — as the foundation for the binary- and configuration-level analysis presented in Sections 15-22.

Illustrative Tests Performed

- Verify the IPA unpacks to the expected Payload/NovaSpherePay.app structure with no unexpected top-level content.
- Verify embedded third-party frameworks are legitimately bundled under Frameworks/ and correspond to a documented SDK dependency.
- Verify PlugIns/ contains only the expected App Extensions (Home Screen Widget, Share/Action Extension, Notification Service Extension, Notification Content Extension) with no unexpected or orphaned extension bundles.
- Verify embedded.mobileprovision and the compiled entitlements are present, internally consistent, and scoped as expected.
- Verify the code signature (_CodeSignature/) covers the full bundle with no unsigned or modified resources.

Illustrative Observations

The extracted IPA presented a standard Payload/NovaSpherePay.app structure with no unexpected top-level content. NovaSpherePay.app/Frameworks/ contains the fictional third-party embedded frameworks itemized in Appendix B (MapSDKKit, AnalyticsKit, CrashReportKit, PartnerLinkKit, ImageCacheKit, and the legacy DocExportKit), each corresponding to a documented SDK dependency; no unexpected or unaccounted-for embedded framework was found. NovaSpherePay.app/PlugIns/ contains exactly the four expected App Extension bundles (the Home Screen Widget, a Share/Action Extension, a Notification Service Extension, and a Notification Content Extension); the widget extension's shared-container and Keychain-sharing configuration is examined further in Sections 16 and 25, and all four App Extension targets are itemized in Appendix F. The embedded.mobileprovision and compiled entitlements.plist are present and internally consistent with the entitlements register reviewed in Section 16, and the _CodeSignature/ manifest covers the full bundle with no unsigned resources identified. No standalone finding is raised against the package structure itself; structure-derived findings are cross-referenced into their respective sections below.

No findings raised against this fictional testing category beyond items cross-referenced elsewhere in this report.

Core IPA Metadata

The table below summarizes the fictional IPA's core package metadata, extracted during static analysis. Full metadata detail appears in Appendix B.

Attribute	Value
IPA File Name	NovaSpherePay-6.4.2.ipa
Bundle Identifier	com.novasphere.pay.ios
Bundle Short Version String / Bundle Version	6.4.2 (build 6402)
Minimum iOS Deployment Target	16.0
Target / Compiled SDK	iOS 18 SDK
Supported Architecture	arm64 (arm64e not observed)
Primary Language / Runtime	Swift 5.x, with a small number of legacy Objective-C bridging headers
Code Signing Scheme	Apple Distribution certificate + App Store provisioning profile (fictional production identity; no real fingerprint disclosed)
Provisioning Profile Type	App Store distribution provisioning profile (fictional)
Entitlements	See Appendix D — Entitlements Register

Attribute	Value
Encryption Export Compliance (ITSAppUsesNonExemptEncryption)	NO — standard HTTPS/TLS exempt encryption only
Symbol Stripping / Obfuscation	Release build strips debug symbols; no name-mangling or control-flow obfuscation applied — see IOS-039
Debuggable Entitlement (get-task-allow)	Not present in the distribution build (confirmed via codesign -d --entitlements)
Build Tool	Xcode 16.x (fictional build configuration), Swift Package Manager for dependency resolution

Embedded / System Framework Inventory

The table below summarizes the frameworks and SDKs linked into the fictional application bundle, distinguishing Apple system frameworks (not embedded — resolved against the OS) from embedded third-party frameworks bundled under Frameworks/. The full inventory, including version and licensing detail, appears in Appendix B.

Framework / SDK	Provider	Usage	Packaging / Cross-Reference
UIKit	Apple System Framework	Primary UI framework	No embedded copy — system-provided
SwiftUI	Apple System Framework	Widget extension and select modern screens	No embedded copy — system-provided
Foundation	Apple System Framework	Core runtime services	No embedded copy — system-provided
Security	Apple System Framework	Keychain Services, SecTrust APIs	No embedded copy — system-provided; see IOS-001, IOS-002, IOS-011
LocalAuthentication	Apple System Framework	Face ID / Touch ID app-unlock gate	No embedded copy — system-provided; see IOS-013
WebKit	Apple System Framework	Support/help WKWebView	No embedded copy — system-provided; see IOS-006
CryptoKit	Apple System Framework	Device-binding nonce and local hashing operations	No embedded copy — system-provided; see IOS-024
UserNotifications	Apple System Framework	Transaction/security alert push notifications	No embedded copy — system-provided; see IOS-019
AuthenticationServices	Apple System Framework	Third-party bank-account aggregation OAuth presentation (ASWebAuthenticationSession)	No embedded copy — system-provided; see IOS-005
MapSDKKit	Fictional Third-Party Vendor A	Branch/ATM-locator map rendering	Embedded framework — see IOS-009 (hardcoded API key)
AnalyticsKit	Fictional Third-Party Vendor B	Usage analytics, funnel tracking	Embedded framework — see IOS-023, Appendix M
CrashReportKit	Fictional Third-Party Vendor C	Crash and non-fatal error reporting	Embedded framework — see IOS-043, Appendix M
PartnerLinkKit	Fictional Third-Party Vendor E	Bank-account aggregation OAuth account-linking flow	Embedded framework — see IOS-005, Appendix M
ImageCacheKit	Fictional Third-Party Vendor D	Remote avatar/promo image loading and caching	Embedded framework — see IOS-031, Appendix M
DocExportKit	Fictional Third-Party Vendor G (legacy)	Legacy statement-export backend client	Embedded framework — see IOS-007, IOS-021, Appendix M

15 INFO.PLIST ANALYSIS

TMG Security reviewed the fictional application's Info.plist in full, covering the bundle identity keys, App Transport Security (ATS) configuration, registered URL types, background modes, and every declared usage-description ('privacy') key, cross-referencing each against the current shipping feature set.

Illustrative Tests Performed

- Verify CFBundleIdentifier, CFBundleShortVersionString, and CFBundleVersion match the release under test.
- Verify NSAppTransportSecurity enforces TLS 1.2+ with no arbitrary-loads exception, and review any per-domain NSExceptionDomains entries individually.
- Verify every declared NS*UsageDescription key corresponds to a capability actually requested by a current, reachable UI flow.
- Verify CFBundleURLTypes entries correspond to actively used custom URL schemes, not legacy or unused handlers.
- Verify ITSAppUsesNonExemptEncryption accurately reflects the app's use of only standard, exempt TLS/HTTPS encryption.

Illustrative Observations

The fictional Info.plist's bundle identity keys correctly match the release under test (com.novasphere.pay.ios, 6.4.2, build 6402). The base NSAppTransportSecurity policy enforces TLS 1.2+ with no arbitrary-loads exception; however, a per-domain NSExceptionDomains entry for the legacy statement-export domain (docs.novasphere-example.com) permits a minimum TLS version of 1.1 (IOS-021), and the base policy applies uniformly to first-party payments-critical domains and lower-sensitivity third-party SDK domains alike with no domain-level granularity (IOS-027). Cross-referencing every declared usage-description key against the current UI walkthrough found NSCameraUsageDescription and NSCalendarsUsageDescription present from a prior release cycle with no reachable feature currently requesting either capability (IOS-028). The full Info.plist key register, including CFBundleURLTypes, background modes, and remaining usage-description keys, is itemized in Appendix C.

Finding ID	Severity	Title
IOS-021	MEDIUM	Insecure ATS Exception Configuration Permits Weak TLS on Secondary Endpoint
IOS-027	MEDIUM	Missing App Transport Security Domain-Level Granularity
IOS-028	LOW	Unused and Unnecessary Privacy-Sensitive Permission Declarations

Info.plist Key Register (Selected)

The table below itemizes the fictional Info.plist keys most relevant to this assessment. The complete extracted key register appears in Appendix C.

Info.plist Key	Value / Configuration	Note	Finding Ref
CFBundleIdentifier	com.novasphere.pay.ios	Application bundle identifier.	—
CFBundleShortVersionString	6.4.2	Marketing version string.	—
CFBundleVersion	6402	Build number.	—
UIRequiredDeviceCapabilities	arm64	Standard 64-bit device capability requirement.	—
NSAppTransportSecurity	Base policy: TLS 1.2+, no arbitrary loads. NSExceptionDomains: docs.novasphere-example.com (NSExceptionMinimumTLSVersion: TLSv1.1)	Legacy statement-export domain exception weakens the TLS floor for one domain; no per-domain granularity for payments-critical domains.	IOS-021, IOS-027
LSApplicationQueriesSchemes	fictionalbankaggregator, fictionalwallet, mapsapp	External app schemes NovaSphere Pay is permitted to query with canOpenURL(_:).	—
CFBundleURLTypes	novasphere, novasphere-support, novasphere-widget, novasphere-legacy, novasphere-promo (see Appendix E for full scheme/handler detail)	Six custom URL scheme entries registered; several correspond to narrow or legacy-only functionality.	IOS-005, IOS-032

Info.plist Key	Value / Configuration	Note	Finding Ref
NSFaceIDUsageDescription	"NovaSphere Pay uses Face ID to securely unlock your account."	Backs the biometric app-unlock gate.	IOS-013
NSCameraUsageDescription	"NovaSphere Pay uses the camera for check deposit."	Declared, but static/dynamic review found no current UI flow that requests camera access.	IOS-028
NSCalendarsUsageDescription	"NovaSphere Pay would like to access your calendar to schedule payment reminders."	Declared, but no current UI flow requests calendar access.	IOS-028
NSContactsUsageDescription	"NovaSphere Pay uses your contacts to help you pay friends."	Requested at first launch; no shipping feature currently references Contacts.	IOS-022
NSMicrophoneUsageDescription	"NovaSphere Pay uses the microphone for voice memo support."	Requested at first launch; no shipping feature currently references the microphone.	IOS-022
NSLocationWhenInUseUsageDescription	"NovaSphere Pay uses your location to find nearby branches and ATMs."	Backs the branch/ATM-locator feature (MapSDKKit); scoped, contextual usage.	—
UIBackgroundModes	remote-notification, fetch	Supports push-triggered background refresh and periodic balance sync.	—
ITSAppUsesNonExemptEncryption	NO	Standard HTTPS/TLS only; no non-exempt (proprietary) encryption shipped in the binary.	—

16 ENTITLEMENTS ANALYSIS

TMG Security extracted and reviewed the fictional application's compiled entitlements — obtained from both the embedded provisioning profile and codesign -d --entitlements against the extracted binary — covering Keychain sharing, Associated Domains, App Groups, and push notification scope.

Illustrative Tests Performed

- Verify keychain-access-groups scope Keychain sharing to only the app targets/extensions that require it.
- Verify com.apple.developer.associated-domains lists only verified, first-party domains and backs Universal Link handling appropriately.
- Verify com.apple.security.application-groups scope shared-container access to only the extensions that require it, with data minimization applied to shared content.
- Verify aps-environment is set appropriately for the release channel (production for the App Store build).
- Verify no unnecessary or overly broad entitlement is present relative to the app's documented feature set.

Illustrative Observations

The compiled entitlements confirm keychain-access-groups scoped to a single shared group ($(AppIdentifierPrefix)com.novasphere.pay.ios.shared) used by both the main app and the Home Screen widget extension; this shared group was found to include the session refresh-token item alongside the widget's own cache data with no narrower, main-app-only group for high-sensitivity session material (IOS-011, detailed further in Section 25). com.apple.developer.associated-domains correctly lists applinks:app.novasphere-example.com, backing the Universal Link handling reviewed in Section 40 of this report's IPC chapter; the silent-commit behavior of the link-payee Universal Link path itself is tracked as IOS-004 rather than an entitlement-configuration defect. com.apple.security.application-groups scopes group.com.novasphere.pay.ios.shared across the main app, the widget extension, and the share extension, and the App Group container's data-minimization gap is tracked as IOS-010. aps-environment is correctly set to production for the release build, consistent with the push-notification content-exposure finding tracked as IOS-019. The full entitlements register appears in Appendix D.

Finding ID	Severity	Title
IOS-011	HIGH	Keychain Access Group Misconfiguration
IOS-004	CRITICAL	Insecure Universal Link / Deep-Link Flow Enables Sensitive Account Action
IOS-010	HIGH	Insecure App Group Data Sharing
IOS-019	MEDIUM	Sensitive Push Notification Data Exposure

Entitlements Register

The table below itemizes the fictional application's compiled entitlements and cross-references each to the findings it underlies. The complete register appears in Appendix D.

Entitlement	Value	Purpose / Note	Finding Ref
keychain-access-groups	$(AppIdentifierPrefix)com.novasphere.pay.ios.shared	Shared Keychain access group between the main app and the Home Screen widget extension.	IOS-002, IOS-011
com.apple.developer.associated-domains	applinks:app.novasphere-example.com	Universal Link verification domain for the beneficiary-linking, statement, support, and promo flows.	IOS-004, IOS-032
com.apple.security.application-groups	group.com.novasphere.pay.ios.shared	Shared container between the main app, the widget extension, and the share extension.	IOS-010, IOS-017
aps-environment	production	Enables APNs push delivery for transaction and security alert notifications.	IOS-019
com.apple.developer.authentication-services.autofill-credential-provider	Not declared	The app does not implement a credential-provider extension; login uses the in-app form only.	—
com.apple.developer.default-data-protection	NSFileProtectionCompleteUntilFirstUserAuthentication	Default Data Protection class applied to the app's on-disk container; several specific files do not raise this to a stricter class — see storage findings.	IOS-001, IOS-026

17 PROVISIONING PROFILE REVIEW

TMG Security extracted the embedded.mobileprovision file from the fictional release IPA and reviewed the provisioning profile's type, entitlement alignment, expiry, and device/team scoping against the entitlements register examined in Section 16.

Illustrative Tests Performed

- Verify the provisioning profile type (development, ad-hoc, App Store distribution, Enterprise) matches the intended release channel.
- Verify the entitlements embedded in the provisioning profile match the compiled entitlements.plist with no discrepancy.
- Verify the provisioning profile has not expired and is not approaching expiry without a documented renewal process.
- Verify device UDID scoping (for ad-hoc/development profiles) or team scoping is appropriate for the release channel under test.
- Verify a documented lifecycle process exists for provisioning profile renewal and revocation.

Illustrative Observations

The embedded provisioning profile is a valid App Store distribution profile, correctly scoped to the fictional NovaSphere production team identifier with no device-UDID list (as expected for an App Store distribution profile) and no fictional Enterprise or ad-hoc distribution entitlement present. The entitlements embedded in the provisioning profile matched the compiled entitlements.plist reviewed in Section 16 with no discrepancy, and the profile's expiry date was confirmed current

as of testing. However, no documented lifecycle process for provisioning profile renewal, rotation, or emergency revocation was identified during this review (IOS-047); this is a process-documentation gap rather than a defect in the current, validly configured profile.

Finding ID	Severity	Title
IOS-047	INFORMATIONAL	Recommendation — Document Provisioning Profile and Signing Certificate Lifecycle Management

18 CODE SIGNING & SIGNATURE VALIDATION

TMG Security validated the fictional release IPA's code signature end to end — the signing certificate chain, the _CodeSignature/ resource manifest, and whether the distributed binary carries any development- or ad-hoc-signing artifact that should not reach a production release.

Illustrative Tests Performed

- Verify the binary's code signature validates successfully against the Apple Distribution certificate chain (codesign --verify --deep --strict).
- Verify the signing certificate chain resolves to a trusted Apple root with no self-signed or expired intermediate.
- Verify the release build is not signed with a development or ad-hoc identity, and does not carry the get-task-allow debuggable entitlement.
- Verify every embedded framework and App Extension in the bundle is independently signed and covered by the outer signature's resource manifest.

Illustrative Observations

Signature validation against the fictional release IPA completed successfully: codesign --verify --deep --strict reported a valid signature for the main executable and every embedded framework and App Extension, and the certificate chain resolved to a valid fictional Apple Distribution certificate with no self-signed or expired intermediate. The distribution build correctly carries an Apple Distribution identity with no development- or ad-hoc-signing artifact, and the get-task-allow debuggable entitlement was confirmed absent, consistent with the metadata reviewed in Appendix B. No standalone finding is raised in this section; the absence of a documented signing lifecycle process is tracked separately as IOS-047 in Section 17.

No findings raised against this fictional testing category beyond items cross-referenced elsewhere in this report.

19 MACH-O BINARY ANALYSIS

TMG Security extracted the main NovaSpherePay Mach-O executable and every embedded framework binary from the fictional release IPA and performed representative otool-/strings-style static binary analysis — architecture and load-command review, symbol table inspection, and a full string-table extraction pass — as the foundation for the source-level review in Section 20.

Illustrative Tests Performed

- Verify the binary architecture matches the intended release target (arm64) with no unexpected slice.
- Verify the symbol table does not disclose more internal structure than expected for a release (non-debug) build.
- Verify every embedded framework binary is accounted for and matches the SDK inventory in Appendix B.
- Verify a strings pass against the binary does not recover hardcoded credentials, API keys, or internal-only endpoint references.

Illustrative Observations

otool -hv confirmed a single arm64 Mach-O slice with no unexpected architecture present, consistent with the fictional release target; no arm64e slice was observed. Symbol table review (nm -a-style extraction) found release-build symbol stripping applied as expected for a distribution build, though readable Swift type and method names remain broadly recoverable, addressed further in Section 20. Every embedded framework binary listed in Appendix B was accounted for in the extracted bundle with no orphaned or unexpected Mach-O binary present. A full strings extraction against the main executable recovered the mapping/geolocation SDK's API key as a plaintext string literal in the __TEXT.__cstring section, requiring no decompilation or runtime instrumentation to obtain (IOS-009). The broader string-table sample, including production and legacy endpoint references, is itemized below and in Appendix B.

Finding ID	Severity	Title
IOS-009	HIGH	Hardcoded API Secret / Credential Material in iOS Binary

Native / Embedded Binary Inventory

The table below lists the Mach-O binaries present in the extracted fictional IPA. The full inventory appears in Appendix B.

Binary	Architecture	Description
NovaSpherePay	Main Mach-O executable (arm64)	Primary application binary.
libnovasphere-crypto.dylib	arm64	Fictional native helper wrapping select CryptoKit/Security operations.
MapSDKKit.framework/MapSDKKit	arm64	Third-party mapping SDK embedded framework binary.
AnalyticsKit.framework/AnalyticsKit	arm64	Third-party analytics SDK embedded framework binary.
CrashReportKit.framework/ CrashReportKit	arm64	Third-party crash-reporting SDK embedded framework binary.
PartnerLinkKit.framework/ PartnerLinkKit	arm64	Third-party bank-aggregation OAuth SDK embedded framework binary.
ImageCacheKit.framework/ ImageCacheKit	arm64	Third-party image loading/caching SDK embedded framework binary.
DocExportKit.framework/DocExportKit	arm64	Legacy third-party statement-export client embedded framework binary.

Selected Extracted Strings

The table below summarizes representative strings recovered from a static extraction pass against the main executable, including the hardcoded mapping SDK key underlying IOS-009. The full extracted string sample appears in Appendix B.

Extracted String	Note
https://api.novasphere-example.com/v6/	Production mobile-API base URL (fictional, non-resolving).
https://auth.novasphere-example.com/oauth/	Production auth/OAuth base URL (fictional, non-resolving).
https://docs.novasphere-example.com/export/	Legacy statement-export document backend URL — see IOS-007, IOS-021 (fictional, non-resolving).
https://staging-api.novasphere-example.com/v6/	Fictional staging endpoint reference retained in the release binary's string table.
MAPS_API_KEY=fictional_maps_sdk_key_8a41cf209b	Hardcoded mapping SDK key — see IOS-009.
novasphere://oauth-callback	Custom URL scheme OAuth callback — see IOS-005.
// TODO: remove QA verbose logging flag before App Store submission	Fictional residual developer comment referencing the verbose-logging configuration tracked in IOS-008.
FICTIONAL_DEBUG_MENU_ENABLED=0	Fictional residual debug-menu compile flag string, confirmed disabled in the release configuration.

20 SWIFT / OBJECTIVE-C STATIC ANALYSIS

TMG Security performed a manual, source-level static review of the fictional application's Swift codebase and its small surface of Objective-C bridging code, focused on security-relevant classes (session management, token handling, cryptography, deep-link and Universal Link handlers) and on whether any obfuscation or code-hardening measure limits static readability.

Illustrative Tests Performed

- Verify security-relevant Swift type and method names are not more readable than a release build should permit (obfuscation posture).
- Verify decompiled/disassembled logic matches the behavior observed in dynamic testing, with no hidden or conditional divergence.
- Verify legacy Objective-C code paths are isolated from the primary Swift codebase and do not carry forward deprecated, insecure API usage into active flows.

Illustrative Observations

Symbol-table and disassembly review confirmed that security-relevant Swift types and methods — including SessionManager, PayeeLinkHandler, and the biometric-vault Keychain wrapper referenced throughout this report's findings — remain fully readable by type and method name, with no symbol renaming, string obfuscation, or control-flow obfuscation applied anywhere in the binary (IOS-039). No evidence of intentionally obfuscated malicious logic, and no dynamic/static behavioral divergence, was identified in any reviewed module. Separately, static review identified an isolated legacy Objective-C networking and UI helper module retaining deprecated, NSURLConnection-era API references alongside the primary, modern URLSession-based networking stack; these deprecated call sites are confined to the legacy module and are not reachable from the app's current primary networking flows (IOS-029).

Finding ID	Severity	Title
IOS-039	INFORMATIONAL	Recommendation — Evaluate Binary Obfuscation for Sensitive Business Logic
IOS-029	LOW	Use of Deprecated iOS APIs With Superseded Secure Alternatives

21 BINARY SECURITY PROTECTIONS

TMG Security verified the fictional release Mach-O binary's platform-level exploit-mitigation protections — stack canaries and Position-Independent Executable (PIE) support — via manual otool-style header and load-command review, and assessed whether these protections are verified automatically as part of the release pipeline.

Illustrative Tests Performed

- Verify the binary is compiled as a Position-Independent Executable (PIE) (otool -hv, MH_PIE flag).
- Verify stack-canary (stack-protector) support is present in the compiled binary.
- Verify these protections are checked automatically for every release build, not only during a point-in-time manual review.

Illustrative Observations

Manual otool -hv and otool -Iv review confirmed the release binary carries the MH_PIE flag and standard stack-canary (___stack_chk_guard / ___stack_chk_fail) support, consistent with current Xcode/Swift toolchain defaults. Both protections were present and correctly configured in the build assessed. However, no automated CI/CD verification step was identified that checks for these flags on every release build; their presence today relies on toolchain defaults confirmed only through this point-in-time manual review, so a future build-configuration change could silently regress either protection without detection (IOS-035).

Finding ID	Severity	Title
IOS-035	LOW	Missing CI Verification of Stack-Canary and PIE Binary Protections

22 DEBUG / RELEASE CONFIGURATION

TMG Security compared the fictional QA/pre-production build configuration against the App Store release configuration under test, focused on whether debug-only behavior — verbose error surfacing, debug flags, and debugging entitlements — is correctly excluded from the distributable release build.

Illustrative Tests Performed

- Verify the release build's DEBUG compilation flag is disabled and no debug-only code path is reachable.
- Verify the get-task-allow debuggable entitlement is absent from the release build (cross-referenced from Section 18).
- Verify error-handling behavior is consistent (generic, user-safe) across every build configuration distributed for QA or production use.
- Verify no build-configuration flag silently re-enables verbose diagnostic output in a build channel closer to production than a pure debug build.

Illustrative Observations

The App Store release build under test correctly disables the DEBUG compilation flag, and the get-task-allow debuggable entitlement is confirmed absent, consistent with the code signature and entitlements review in Sections 16 and 18. However, the fictional QA/pre-production build configuration — used for pre-release validation and closer to production than a pure debug build — retains a build-flag-gated debug alert path in the beneficiary-management error-handling flow that surfaces the raw backend error response, including an internal service name and a partial ORM query fragment, directly to the screen rather than the generic error presentation used in the App Store build (IOS-025). This is a QA-build-configuration gap rather than a defect in the production release build itself, which correctly uses generic error handling throughout.

Finding ID	Severity	Title
IOS-025	MEDIUM	Verbose Error Handling Reveals Internal Implementation Details

23 LOCAL STORAGE SECURITY

ILLUSTRATIVE / SYNTHETIC EVIDENCE
no real device, account, or data

```
ssh mobile@fictional-test-device-a

$ ssh mobile@fictional-test-device-a
iPhone-Fictional:~ mobile$ cd .../Data/Application/<FICTIONAL-GUID>/Library/Preferences
iPhone-Fictional:Preferences mobile$ cat com.novasphere.pay.ios.session.plist

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "...PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>access_token</key>
    <string>eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.FICTIONAL_PAYLOAD.sig</string>
    <key>refresh_token</key>
    <string>rt_novasphere_fictional_7c1e9b40aa2f</string>
    <key>device_binding_secret</key>
    <string>db_fictional_ios_51f0ac3d</string>
    <key>token_issued_at</key>
    <integer>1799999999</integer>
</dict>
</plist>

# Result: tokens recovered in CLEARTEXT — NSUserDefaults plist, no Keychain barrier
```

Finding IOS-001 — Sensitive Authentication Token Exposure Through Insecure Local Storage

Fictional NSUserDefaults Preferences plist extraction — ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE.

TMG Security extracted and reviewed every fictional non-Keychain local-storage surface NovaSphere Pay writes to — NSUserDefaults-backed preference plists and the offline transaction-history cache — on a fictional jailbroken test device, to assess whether sensitive values are protected at rest outside the Keychain-specific review in Sections 24-25.

Illustrative Tests Performed

- Verify session/access tokens are not stored in NSUserDefaults or any other non-Keychain plist-backed storage.
- Verify refresh tokens and device-binding secrets are not stored outside the Keychain.
- Verify locally cached transaction data applies an appropriate Data Protection class or equivalent encryption at rest.
- Verify no sensitive value is written to Application Support or Documents without a Data Protection class stricter than the app-wide default where warranted.

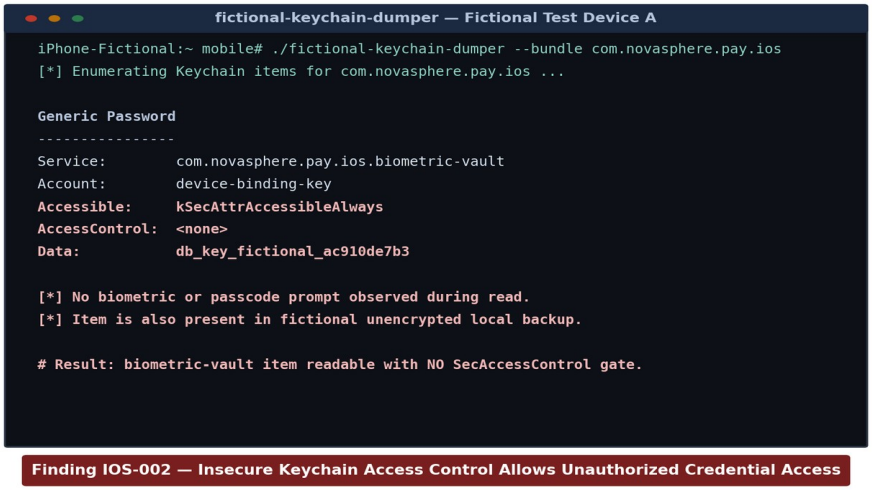
Illustrative Observations

The com.novasphere.pay.ios.session NSUserDefaults suite stores the OAuth access token, refresh token, and a fictional device-binding secret in a standard, unencrypted Preferences plist, recoverable directly from the app's container on a fictional jailbroken test device with no Keychain-backed cryptographic barrier (IOS-001). This is the most severe finding in this fictional assessment given its direct path to fictional account takeover; full technical detail, including the recovered plist contents, is presented in the corresponding finding card. Separately, the offline transaction-history cache used to support browsing recent activity without connectivity is persisted as a plain Codable-encoded JSON file (transaction_cache.json) in the app's Application Support directory at the default NSFileProtectionCompleteUntilFirstUserAuthentication class with no additional encryption, and was similarly recoverable in cleartext from the fictional jailbroken test device (IOS-026).

Finding ID	Severity	Title
IOS-001	CRITICAL	Sensitive Authentication Token Exposure Through Insecure Local Storage
IOS-026	MEDIUM	Insecure Object Persistence of Sensitive Local Cache

24 KEYCHAIN SECURITY

ILLUSTRATIVE / SYNTHETIC EVIDENCE
no real device, account, or data



Fictional Keychain dump — biometric-vault item — ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE.

TMG Security enumerated the fictional application's Keychain items on a jailbroken test device and reviewed the accessibility class and access-control policy applied to each, focused on the Keychain items that back the biometric-unlock and device-binding features.

Illustrative Tests Performed

- Verify Keychain items protecting sensitive credential or key material use a device-locked accessibility class (kSecAttrAccessibleWhenUnlockedThisDeviceOnly or stricter).
- Verify Keychain items backing biometric re-authentication are additionally protected by a SecAccessControl requiring current biometry enrollment.
- Verify no Keychain item uses kSecAttrAccessibleAlways or another unrestricted accessibility class for sensitive material.
- Verify sensitive Keychain items are correctly excluded from unencrypted device backups where the accessibility class supports it.

Illustrative Observations

A fictional Keychain enumeration on a jailbroken test device found the com.novasphere.pay.ios.biometric-vault item — which wraps the long-lived device-binding key used to re-establish an authenticated session — stored with kSecAttrAccessibleAlways and no SecAccessControl at all, rather than a device-locked accessibility class combined with a biometry-bound access-control policy. The item's plaintext value was recoverable with no biometric or passcode challenge and was also present in a fictional unencrypted local backup (IOS-002). This is the second Critical finding in this fictional assessment: because the item is not bound to biometry, the intended Face ID gate protecting session re-establishment provides no actual cryptographic barrier at the Keychain layer. All other Keychain items reviewed used an appropriate device-locked accessibility class; no additional accessibility-class weakness was found outside the biometric-vault item.

Finding ID	Severity	Title
IOS-002	CRITICAL	Insecure Keychain Access Control Allows Unauthorized Credential Access

25 KEYCHAIN ACCESS GROUPS

TMG Security reviewed how NovaSphere Pay's Keychain items are partitioned across keychain-access-groups entitlements between the main app and the Home Screen widget extension, distinct from the per-item accessibility-class review in Section 24.

Illustrative Tests Performed

- Verify each keychain-access-groups entry is scoped to the minimum set of targets/extensions that require it.
- Verify high-sensitivity session material (access/refresh tokens, device-binding keys) is not provisioned into a Keychain access group shared with an App Extension.
- Verify an App Extension's Keychain access is limited to only the items it functionally requires.

Illustrative Observations

The fictional application provisions a single shared Keychain access group ($(AppIdentifierPrefix)com.novasphere.pay.ios.shared) covering both the main app and the Home Screen widget extension, rather than splitting session material into a narrower, main-app-only access group. As a result, the session refresh-token Keychain item and the widget extension's own cached-balance display item both reside in the same access group, so any code executing within the widget extension's process context — including a future compromised or overly broad third-party widget dependency — can query the shared access group and retrieve the refresh token alongside the widget's own data (IOS-011). This finding is distinct from the item-level accessibility weakness on the biometric-vault item (IOS-002, Section 24): here the affected item's accessibility class is otherwise appropriate, but its access-group scope is broader than the widget extension's actual functional need.

Finding ID	Severity	Title
IOS-011	HIGH	Keychain Access Group Misconfiguration

26 CRYPTOGRAPHIC IMPLEMENTATION

TMG Security reviewed NovaSphere Pay's use of cryptography against OWASP MASVS-CRYPTO — algorithm and mode selection via CryptoKit, key handling for local field-level encryption, and the randomness source used for every security-relevant generated value, including the device-binding nonce produced during fictional device registration.

Illustrative Tests Performed

- Verify local field-level encryption uses a current, authenticated algorithm and mode (e.g., AES-GCM) via CryptoKit rather than a legacy CommonCrypto call with an unauthenticated mode.
- Verify initialization vectors, salts, and symmetric keys used in CryptoKit operations are generated fresh per operation.
- Verify no cryptographic key is hardcoded in application code or resources (distinct from the hardcoded service credential tracked as IOS-009).
- Verify every security-relevant random value — cryptographic and non-cryptographic alike — is generated using SecRandomCopyBytes or an equivalent CSPRNG.

Illustrative Observations

NovaSphere Pay's CryptoKit usage was found sound overall: local field-level encryption of select cached values uses AES-GCM with a fresh, CryptoKit-generated symmetric key and nonce per operation, and no use of a legacy unauthenticated mode (e.g., CommonCrypto AES-ECB) or a hardcoded cryptographic key was identified anywhere in the reviewed codebase. The one confirmed weakness in this category is narrower than a systemic cryptographic-implementation failure: the device-binding nonce generated during fictional device registration is derived using arc4random_uniform combined with a value influenced by device boot time, rather than SecRandomCopyBytes, the platform's CSPRNG intended for security-relevant randomness. This produces a nonce with a narrower, more predictable generation window than intended (IOS-024). Because the device-binding nonce contributes to, but is not solely sufficient to bypass, the device-registration check, this finding is scored Medium rather than Critical/High severity, consistent with its treatment elsewhere in this report.

Finding ID	Severity	Title
IOS-024	MEDIUM	Insecure Random Number Generation for a Security-Relevant Value

27 AUTHENTICATION

TMG Security reviewed NovaSphere Pay's overall login flow — credential submission, multi-factor enforcement at login, lockout behavior, and the account-recovery path that re-establishes trust when a user loses access to their original device — against OWASP MASVS-AUTH.

Illustrative Tests Performed

- Verify login credentials are transmitted only over the TLS-protected primary API channel (cross-referenced from Section 31).
- Verify multi-factor authentication is enforced at login and cannot be bypassed by a direct API call that skips the MFA-verification step.
- Verify account lockout / rate-limiting is applied after repeated failed authentication attempts.
- Verify account recovery requires re-verification through a channel independent of the potentially compromised device and cannot skip MFA re-enrollment.

Illustrative Observations

NovaSphere Pay correctly enforces multi-factor authentication at login, and TMG Security's fictional testing confirmed the MFA-verification step cannot be bypassed by calling the session-issuance endpoint directly without a validated one-time-code assertion. Failed-attempt lockout is applied consistently. Account recovery independently requires two out-of-band factors (email and SMS one-time codes) before a credential reset is permitted, and mandatory MFA re-enrollment cannot be skipped during recovery — this is recorded as a positive control observation (IOS-036). No findings are raised against this

testing area; the local-authentication weakness affecting the app-unlock gate is treated separately in Section 28, and the object-level authorization weakness identified in the mobile API surface is treated separately in Section 36.

Finding ID	Severity	Title
IOS-036	INFORMATIONAL	Positive Observation — Multi-Factor Authentication Enforced on Account Recovery

28 BIOMETRIC AUTHENTICATION / FACE ID

TMG Security reviewed NovaSphere Pay's Face ID app-unlock feature (LocalAuthentication) for correct binding to an underlying cryptographic operation, and separately assessed whether the highest-value transaction type warrants a transaction-scoped biometric checkpoint beyond the app-unlock gate.

Illustrative Tests Performed

- Verify LAContext.evaluatePolicy is used to gate a Keychain operation protected by a biometry-bound SecAccessControl, rather than as a bare presentation-layer boolean check.
- Verify a successful Face ID prompt cannot be spoofed by directly forcing the completion handler's result.
- Verify the app falls back to device passcode rather than silently unlocking when biometric hardware is unavailable or unenrolled.
- Verify high-value transfers are protected by a transaction-scoped step-up authentication check beyond standard app-unlock biometrics.

Illustrative Observations

NovaSphere Pay's Face ID app-unlock gate checks the boolean success result returned by LAContext.evaluatePolicy's completion handler, but that result is not cryptographically anchored to a Keychain item protected by a matching SecAccessControl. TMG Security's fictional instrumentation testing hooked the completion handler to force a success result and bypassed the Face ID gate entirely without providing any valid biometric input (IOS-013). Device-passcode fallback behavior when biometric hardware is unavailable functions correctly and is not itself weakened by this finding. Separately, TMG Security observed that biometric authentication is gated only once at app unlock — a session-active user can initiate a high-value transfer without a second, transaction-scoped biometric confirmation. TMG Security recommends the app adopt step-up biometric re-authentication for transfers above a defined value threshold as a defense-in-depth enhancement (IOS-041).

Finding ID	Severity	Title
IOS-013	HIGH	Biometric Authentication Bypass / Improper LocalAuthentication Enforcement
IOS-041	INFORMATIONAL	Recommendation — Add Step-Up Biometric Re-Authentication for High-Value Transactions

29 SESSION MANAGEMENT

TMG Security reviewed NovaSphere Pay's session lifetime, idle-timeout enforcement, and whether session validity is independently enforced server-side rather than relying solely on client-side state.

Illustrative Tests Performed

- Verify an idle-timeout value is enforced and re-authentication is required after it elapses.
- Verify the enforced idle-timeout value is consistent with the sensitivity of a financial application and is independently configurable for higher-risk fictional account tiers.
- Verify session validity is independently checked server-side (session age), not only through a client-side idle timer.

Illustrative Observations

An idle-timeout is enforced client-side, re-prompting for local authentication after the app has been backgrounded past the configured interval. However, the configured value (45 minutes) is longer than appropriate for a financial application

handling live account and payment data, is applied uniformly regardless of account risk tier, and enforcement is entirely client-side — no corresponding server-side session-age check was observed during testing (IOS-020). The related refresh-token revocation gap affecting the end of the session lifecycle is tracked separately as IOS-014 in Section 30.

Finding ID	Severity	Title
IOS-020	MEDIUM	Weak Session Timeout Enforcement

30 TOKEN LIFECYCLE

TMG Security reviewed the full lifecycle of NovaSphere Pay's OAuth 2.0 access and refresh tokens — issuance, expiry, and revocation on logout — extending the local-storage findings from Section 6 with a lifecycle-management focus.

Illustrative Tests Performed

- Verify access and refresh tokens are stored using an appropriate protection mechanism (cross-referenced: IOS-001, IOS-002).
- Verify the refresh token is revoked server-side when the user explicitly logs out.
- Verify access tokens carry an appropriately short expiry consistent with the sensitivity of the operations they authorize.

Illustrative Observations

The storage-at-rest weaknesses affecting the access token, refresh token, and biometric-vault device-binding key are tracked as IOS-001 and IOS-002 and are not re-scored here. Independently, TMG Security's fictional testing found that when a user logs out, the client removes the local Keychain-resident refresh token but the fictional NovaSphere Enterprise Platform API does not receive or process any explicit token-revocation call — a refresh token captured prior to logout remained valid and could be replayed to obtain a new access token well after the user believed they had signed out (IOS-014). This extends the practical exposure window of any prior token-capture path in this report. Access-token expiry (15 minutes) is appropriately short.

Finding ID	Severity	Title
IOS-014	HIGH	Missing Refresh Token Revocation After Logout

31 NETWORK SECURITY

TMG Security reviewed NovaSphere Pay's transport-layer security posture at an overview level — TLS configuration strength, App Transport Security (ATS) scoping, and certificate/trust handling — against OWASP MASVS-NETWORK, ahead of the detailed treatment of each area in Sections 32-34.

Illustrative Tests Performed

- Verify the minimum negotiable TLS version is 1.2 across all first-party endpoints.
- Verify only strong, forward-secret cipher suites are offered by the client.
- Verify no automatic fallback to a legacy TLS/SSL version occurs on handshake failure.

Illustrative Observations

The primary URLSession-based API client, used for authentication, account, and payment traffic against api.novasphere-example.com and auth.novasphere-example.com, correctly enforces TLS 1.2/1.3 as a floor with strong, forward-secret cipher suites and rejects downgrade attempts — recorded as a positive control observation (IOS-037). A secondary, older networking path retained for the statement-export subsystem's legacy backend integration carries the weaker TLS configuration and certificate-validation posture detailed in Sections 32 and 33.

Finding ID	Severity	Title
IOS-037	INFORMATIONAL	Positive Observation — Strong TLS Configuration on Primary API Endpoints

32 APP TRANSPORT SECURITY

TMG Security reviewed NovaSphere Pay's Info.plist NSAppTransportSecurity (ATS) configuration for domain scoping, minimum-TLS-version exceptions, and overall policy granularity.

Illustrative Tests Performed

- Verify ATS does not carry an NSExceptionDomains entry permitting a minimum TLS version below 1.2 for any first-party domain.
- Verify ATS applies materially stricter requirements to payments-critical domains (api.novasphere-example.com, auth.novasphere-example.com) than to lower-sensitivity third-party SDK domains.
- Verify no blanket NSAllowsArbitraryLoads exception is present in the release build configuration.

Illustrative Observations

No blanket NSAllowsArbitraryLoads exception is present in the release configuration. However, the ATS configuration carries an NSExceptionDomains entry for the legacy statement-export domain (docs.novasphere-example.com) permitting a minimum TLS version of 1.1, rather than the 1.2+ floor enforced for all other first-party domains (IOS-021). More broadly, ATS applies one shared baseline policy to all domains — first-party payments-critical endpoints and third-party SDK endpoints alike — rather than scoping stricter requirements specifically to the payments-critical hosts, reducing the precision of the app's trust boundaries (IOS-027).

Finding ID	Severity	Title
IOS-021	MEDIUM	Insecure ATS Exception Configuration Permits Weak TLS on Secondary Endpoint
IOS-027	MEDIUM	Missing App Transport Security Domain-Level Granularity

33 TLS VALIDATION

TMG Security tested NovaSphere Pay's certificate-chain and trust-evaluation logic under fictional proxy-interception conditions to confirm the application correctly rejects an untrusted or improperly chained certificate rather than silently accepting it.

Illustrative Tests Performed

- Verify every custom URLSessionDelegate trust-evaluation callback enforces the result of SecTrustEvaluateWithError (or equivalent) as a hard gate before accepting a server credential.
- Verify the application rejects a self-signed or otherwise untrusted certificate presented by an interception proxy under default device trust settings.
- Verify no custom delegate unconditionally calls completionHandler(.useCredential, ...) regardless of trust-evaluation outcome.

Illustrative Observations

A custom URLSessionDelegate implementation used only by the legacy statement-export module (the same client identified in Sections 31-32) contains a certificate-chain validation defect: the delegate calls SecTrustEvaluateWithError and logs the result, but does not enforce it as a pass/fail gate before calling completionHandler(.useCredential, ...). Under a fictional proxy-interception condition involving an intermediate-certificate substitution, the custom logic accepted the connection where platform-default SecTrust evaluation would have rejected it (IOS-007). The primary API client's certificate validation, which relies on platform-default trust evaluation, was not affected.

Finding ID	Severity	Title
IOS-007	HIGH	Weak Certificate Validation / Trust Evaluation Logic

34 CERTIFICATE PINNING

TMG Security reviewed NovaSphere Pay's certificate/public-key pinning coverage against OWASP MASVS-NETWORK-2, focused on whether pinning is applied to the sensitive first-party payment and authentication endpoints and how pinning coverage relates to the trust-evaluation defect identified in Section 33.

Illustrative Tests Performed

- Verify certificate or public-key pinning is implemented for the primary payments API endpoint.
- Verify pinning failure results in connection termination rather than a silent fallback to unpinned validation.
- Verify the legacy statement-export client's lack of pinning is consistent with, and does not independently expand, the trust-evaluation weakness already tracked against it.

Illustrative Observations

Pinning is correctly implemented for api.novasphere-example.com and auth.novasphere-example.com with a primary and backup pin, and pinning failure correctly terminates the connection rather than falling back to unpinned validation. The legacy statement-export client covered in Sections 32-33 does not implement pinning at all; this is consistent with, and does not independently expand, the IOS-007 and IOS-021 findings already tracked against that client — no additional finding is raised in this section. TMG Security notes that extending pinning to the statement-export domain would provide a partial compensating control for the underlying trust-evaluation defect, though the recommended primary fix remains correcting the delegate itself (see Section 33).

No findings raised against this fictional testing category beyond items cross-referenced elsewhere in this report.

35 API COMMUNICATION

TMG Security enumerated and tested every mobile-API endpoint reachable from NovaSphere Pay's iOS client against the fictional NovaSphere Enterprise Platform API (api.novasphere-example.com, auth.novasphere-example.com), covering authentication, account, transfer, payee, card, profile, and support-channel operations.

Illustrative Tests Performed

- Enumerate every mobile-API endpoint invoked by the iOS client via static and dynamic (proxy-based) analysis.
- Record the authentication requirement, data sensitivity, and purpose of each endpoint.
- Cross-reference each endpoint against this report's findings to confirm every endpoint-level weakness is captured and traceable.

Illustrative Observations

TMG Security's fictional testing enumerated 27 mobile-API endpoints across six functional areas: authentication/token lifecycle, account and statement retrieval, funds transfer, payee management, card detail retrieval, profile/notification management, and the support/telemetry channel. The full register, including each endpoint's authentication requirement, illustrative sensitivity rating, and any associated finding, is presented in the table below. The most severe endpoint-level weakness identified — a broken object-level authorization condition on the funds-transfer confirmation endpoint — is presented with full technical detail in Section 36.

No findings raised against this fictional testing category beyond items cross-referenced elsewhere in this report.

Mobile API Endpoint Register (Appendix G)

The following register reflects the complete illustrative set of mobile-API endpoints exercised during this fictional assessment.

Domain	Method	Endpoint	Auth	Sensitivity	Description	Finding Ref.
auth.novasphere-example.com	POST	/oauth/token	None	Critical	Credential login; issues access and refresh tokens.	—
auth.novasphere-example.com	POST	/oauth/token/refresh	Refresh Token	Critical	Access-token refresh.	—
auth.novasphere-example.com	POST	/oauth/revoke	Session	High	Token revocation on logout.	IOS-014
auth.novasphere-example.com	POST	/oauth/mfa/verify	Partial Session	Critical	MFA one-time-code verification.	—
auth.novasphere-example.com	POST	/oauth/biometric/register	Session	High	Registers the biometric-unlock device-binding key.	IOS-002
auth.novasphere-example.com	POST	/oauth/link/callback	State/ Nonce (expected)	Critical	Bank-account aggregation OAuth account-linking callback.	IOS-005
api.novasphere-example.com	GET	/v6/mobile/accounts	Session	High	Account list for the session owner.	—
api.novasphere-example.com	GET	/v6/mobile/accounts/{accountId}	Session	High	Account detail retrieval.	—
api.novasphere-example.com	GET	/v6/mobile/accounts/{accountId}/statements	Session	High	Statement listing retrieval.	—
api.novasphere-example.com	GET	/v6/mobile/statements/{statementId}/export	Session	High	PDF statement export generation (legacy DocExportKit backend).	IOS-007, IOS-016, IOS-021
api.novasphere-example.com	POST	/v6/mobile/transfers	Session	Critical	Funds-transfer initiation.	IOS-003
api.novasphere-example.com	POST	/v6/mobile/transfers/confirm	Session	Critical	Funds-transfer confirmation — object-level authorization gap.	IOS-003
api.novasphere-example.com	GET	/v6/mobile/transfers/{transferId}	Session	High	Transfer status retrieval.	—
api.novasphere-example.com	GET	/v6/mobile/payees	Session	High	Payee list retrieval.	—
api.novasphere-example.com	POST	/v6/mobile/payees	Session	Critical	Payee creation — reachable silently via Universal Link auto-confirm.	IOS-004
api.novasphere-example.com	DELETE	/v6/mobile/payees/{payeeId}	Session	High	Payee removal.	—
api.novasphere-example.com	GET	/v6/mobile/cards	Session	High	Linked card list retrieval.	—
api.novasphere-example.com	GET	/v6/mobile/cards/{cardId}/detail	Session	Critical	Full (client-masked) card detail retrieval.	IOS-015
api.novasphere-example.com	GET	/v6/mobile/profile	Session	Medium	User profile retrieval.	—
api.novasphere-example.com	PATCH	/v6/mobile/profile	Session	Medium	User profile update.	—
api.novasphere-example.com	POST	/v6/mobile/devices/push-token	Session	Low	Device/push-token registration.	IOS-019
api.novasphere-example.com	POST	/v6/mobile/biometric-challenge	Session	High	Server-side biometric step-up challenge/response (not currently invoked for high-value transfers).	IOS-041
api.novasphere-example.com	GET	/v6/mobile/notifications/preferences	Session	Low	Notification preference retrieval.	—

Domain	Method	Endpoint	Auth	Sensitivity	Description	Finding Ref.
api.novasphere-example.com	GET	/v6/mobile/support/articles	None	Low	Static support-article listing.	—
api.novasphere-example.com	POST	/v6/mobile/support/session-token	Session	Medium	Support-WKWebView session token issuance.	IOS-006
api.novasphere-example.com	POST	/v6/mobile/telemetry/events	Device Token	Low	AnalyticsKit telemetry event ingestion.	IOS-023
api.novasphere-example.com	GET	/v6/mobile/config/feature-flags	None	Low	Feature-flag configuration retrieval.	—

Table 35.1 — Fictional Mobile API Endpoint Register (27 endpoints tested)

36 MOBILE API AUTHORIZATION

TMG Security tested every account-scoped and payment-scoped mobile-API endpoint for object-level authorization enforcement, submitting requests referencing account identifiers other than the authenticated fictional test session's own. This section presents the full technical detail of this report's most severe API-layer finding.

Illustrative Tests Performed

- Verify the funds-transfer confirmation endpoint validates that the client-supplied sourceAccountId belongs to the authenticated session.
- Verify the statement-retrieval endpoint cannot be used to retrieve another fictional account's statement by identifier substitution.
- Verify the payee-management endpoints enforce account-ownership checks server-side, not only in client-side UI logic.

Illustrative Observations

TMG Security's fictional testing identified a Critical broken object-level authorization condition in the funds-transfer confirmation workflow. The POST /v3/mobile/transfers/confirm endpoint accepts a sourceAccountId parameter from the client and processes the fictional transfer against that account without validating server-side that it belongs to the authenticated session. In TMG Security's fictional proof-of-concept, an authenticated iOS session for Test Account A successfully confirmed a transfer sourced from Test Account B's balance by substituting the account identifier in the request body — the platform's most sensitive operation was authorized using nothing more than a client-supplied value (IOS-003). This is the single most severe finding identified against the mobile API surface in this assessment: it requires no privilege escalation, no device compromise, and no social engineering — only a standard authenticated session and knowledge of a target account identifier — and it maps directly to direct fictional financial loss. The statement-retrieval and payee-management endpoints tested correctly enforced server-side ownership checks and did not exhibit the same weakness.

Finding ID	Severity	Title
IOS-003	CRITICAL	Broken Authorization in Mobile API Workflow

37 WKWEBVIEW SECURITY

TMG Security reviewed NovaSphere Pay's single WKWebView instance (the in-app support/help screen) for navigation policy, WKNavigationDelegate enforcement, and JavaScript-execution configuration, consistent with OWASP MASVS-PLATFORM-2.

Illustrative Tests Performed

- Verify WKWebView navigation is restricted to an allow-listed first-party domain set via WKNavigationDelegate decision handling.
- Verify JavaScript execution (WKWebViewConfiguration.preferences.javaScriptEnabled) is enabled only where the loaded content strictly requires it.
- Verify mixed-content and file:// URL loading are disabled in the WKWebView configuration.
- Verify any content injected into the WKWebView is not attacker-influenceable without validation.

Illustrative Observations

The support WKWebView does not restrict navigation to a strict first-party allow-list at the WKNavigationDelegate layer, widening the population of content that could reach the JavaScript bridge assessed in Section 38 (IOS-006, contributing factor). file:// URL loading was confirmed disabled and no mixed-content issue was observed. JavaScript execution is required for the support experience and remains enabled, making the navigation-policy gap the primary compensating control missing from an otherwise conventional WKWebView configuration.

Finding ID	Severity	Title
IOS-006	HIGH	WKWebView JavaScript Bridge Exposes Sensitive Native Functionality

38 JAVASCRIPT BRIDGE SECURITY

TMG Security enumerated and tested every method exposed through NovaSphere Pay's WKScriptMessageHandler bridge (novaBridge), registered via WKUserContentController.add(_:name:) and reachable from the support WKWebView reviewed in Section 37.

Illustrative Tests Performed

- Verify WKScriptMessageHandler methods do not expose sensitive session data to WebView-side JavaScript.
- Verify bridge methods independently verify the calling frame's origin (message.frameInfo.securityOrigin) before responding.
- Verify only the minimum necessary native functionality is exposed through the bridge.

Illustrative Observations

The novaBridge WKScriptMessageHandler exposes getSessionToken and getDeviceIdentifier methods with no origin verification performed by the handler, allowing any JavaScript executing in the WebView context — not only the intended first-party support content — to retrieve the live session token and device identifier by calling window.webkit.messageHandlers.novaBridge.postMessage(...) directly. Combined with the navigation-policy gap noted in Section 37, this widens the population of content able to reach a live-credential-exposing native bridge method (IOS-006). This is this report's sole WKWebView/JavaScript-bridge finding and is rated High given the direct session-token exposure path, achievable without any native code execution.

Finding ID	Severity	Title
IOS-006	HIGH	WKWebView JavaScript Bridge Exposes Sensitive Native Functionality

39 CUSTOM URL SCHEMES

TMG Security reviewed NovaSphere Pay's registered custom URL scheme(s) and the application(_:open:options:) handler(s) that process them, focused on whether any security-sensitive flow relies on a scheme that iOS does not exclusively reserve to one app.

Illustrative Tests Performed

- Verify security-sensitive redirect flows (e.g., OAuth callbacks) use Universal Links rather than an unreserved custom URL scheme.

- Verify any custom-scheme callback independently validates a state/nonce parameter server-side, regardless of transport.
- Verify the number of registered custom URL scheme entries is limited to what current functionality strictly requires.

Illustrative Observations

NovaSphere Pay registers the novasphere:// custom URL scheme, which iOS permits any installed app to also claim. The novasphere://oauth-callback path, used to complete a fictional third-party bank-account aggregation OAuth flow, accepts and processes an authorization code from any caller with no observed state/nonce validation and no verification that the callback originated from the legitimate flow. TMG Security's fictional test harness registered the same scheme and successfully received a simulated OAuth callback intended for NovaSphere Pay, demonstrating that a fictional malicious app claiming the same scheme could intercept or inject a crafted callback (IOS-005). Separately, static review found six registered custom URL scheme entries in total, several corresponding to narrow or legacy-only functionality — TMG Security recommends a periodic surface-area review to retire unused schemes as a hardening measure independent of the OAuth-callback finding (IOS-032).

Finding ID	Severity	Title
IOS-005	HIGH	Insecure Custom URL Scheme Enables Authentication or Account Flow Hijacking
IOS-032	LOW	Attack-Surface Hardening Opportunity — Excessive Registered URL Scheme and Extension Count

40 UNIVERSAL LINKS

TMG Security reviewed NovaSphere Pay's Universal Link handling, implemented via NSUserActivity continuation and the application(_:continue:restorationHandler:) delegate method, for the mobile-specific risk of a link-originated flow committing a sensitive account change with no user confirmation.

Illustrative Tests Performed

- Verify the apple-app-site-association file correctly scopes Universal Link paths to the intended associated domain.
- Verify any Universal Link path that results in a sensitive account change presents an in-app confirmation step before the change is committed.
- Verify no Universal Link parameter can bypass an intended confirmation step (e.g., via an auto-confirm flag).

Illustrative Observations

TMG Security identified a Critical weakness in the Universal Link handler for the /link-payee path under the app.novasphere-example.com associated domain. The NSUserActivity continuation handler parses payee parameters directly from the URL and calls the payee-creation API immediately, with no confirmation UI and no re-authentication step, even though adding a payee is a state-changing, security-relevant action. A fictional crafted Universal Link containing an auto=1 parameter, delivered via a simulated phishing message or malicious QR code, silently added a new payee under attacker control to an authenticated victim's account the moment the link was opened, with no confirmation screen shown at any point (IOS-004). This is one of this report's four Critical findings: it requires only that an authenticated victim tap a single link, removes the last user-facing checkpoint before a follow-on social-engineering-driven funds transfer, and is presented with full technical detail here given its severity and the ease with which the auto-confirm fast-path could be reached.

Finding ID	Severity	Title
IOS-004	CRITICAL	Insecure Universal Link / Deep-Link Flow Enables Sensitive Account Action

41 DEEP LINKS

Sections 39 and 40 assessed NovaSphere Pay's custom URL scheme (IOS-005) and Universal Link (IOS-004) findings individually. This section reviews deep-link handling generally across both transports — parameter validation practice, how

link-originated requests verify the source of the communication channel, and the aggregate deep-link/extension attack surface represented by the app's full URL scheme and extension footprint (Appendix E).

Illustrative Tests Performed

- Verify parameters extracted from an incoming deep link (custom scheme or Universal Link) are treated as untrusted input and validated before use, rather than trusted implicitly because the link opened successfully.
- Verify deep-link handlers that trigger a network call independently re-derive any sensitive identifier (account, payee, transfer reference) server-side rather than trusting the value as parsed from the URL.
- Verify the app does not infer the trustworthiness of a calling context from the mere fact that iOS handed it a URL to open.
- Verify the aggregate count and current-usage status of registered URL schemes and App Extension targets, consolidating the individual register in Appendix E.

Illustrative Observations

Across the six registered custom URL scheme entries and the Universal Link paths enumerated in Appendix E, parameter parsing itself was generally sound — query-string values were percent-decoded and type-checked before use, and TMG Security did not identify an injection or parser-crash condition in any handler tested. The two sensitive-action weaknesses already tracked individually — the silent payee-linking Universal Link flow (IOS-004) and the unreserved OAuth custom-scheme callback (IOS-005) — remain this chapter's most significant deep-link findings and are not re-scored here. Beyond those two, TMG Security's consolidated review of the full URL scheme and extension register found several entries (novasphere-legacy://, novasphere-promo://, and the legacy novasphere://add-payee alias) with narrow or no current first-party usage, broadening the externally reachable attack surface beyond what current functionality requires; this aggregate observation is tracked as IOS-032. No additional individually exploitable deep-link weakness was identified beyond IOS-004, IOS-005, and IOS-032.

Finding ID	Severity	Title
IOS-004	CRITICAL	Insecure Universal Link / Deep-Link Flow Enables Sensitive Account Action
IOS-005	HIGH	Insecure Custom URL Scheme Enables Authentication or Account Flow Hijacking
IOS-032	LOW	Attack-Surface Hardening Opportunity — Excessive Registered URL Scheme and Extension Count

42 APP EXTENSIONS

NovaSphere Pay ships four App Extension targets, enumerated in Appendix F: a Home Screen Widget (WidgetKit), a Share/Action Extension, a Notification Service Extension, and a Notification Content Extension. Each extension runs in its own process with its own, typically more constrained, memory and entitlement context, but still executes reviewable code with access to whatever data the host app chooses to make available to it. TMG Security reviewed each extension's role and the scope of data it is handed relative to its stated task.

Illustrative Tests Performed

- Verify each App Extension receives only the minimum data needed for its specific task, rather than broad read access to a shared cache intended for the main app's internal use.
- Verify extension-to-main-app data handoff uses a scoped payload (e.g., an NSExtensionItem user-info dictionary) rather than a shared file the extension opens independently.
- Verify no App Extension target holds an entitlement or Keychain access group broader than its documented function requires.
- Verify the Notification Service/Content Extensions do not have App Group or Keychain access beyond what their rendering/redaction role requires.

Illustrative Observations

TMG Security's review of the four registered extension targets found: NovaSpherePay Widget (Home Screen Widget Extension (WidgetKit)) — group.com.novasphere.pay.ios.shared — cached account balance, last-four card digits. Shares the same Keychain access group as the main app's refresh token (IOS-011); shared App Group cache is unencrypted (IOS-010). NovaSpherePay Share Extension (Action / Share Extension) — group.com.novasphere.pay.ios.shared — full cached

transaction-history file (not scoped to the shared item). Reads the entire cached transaction-history file rather than a scoped single-transaction payload. NovaSpherePay Notification Service Extension (Notification Service Extension (UNNotificationServiceExtension)) — Push payload delivered by APNs — no App Group data read. Not currently used to redact sensitive transaction detail before lock-screen display. NovaSpherePay Notification Content Extension (Notification Content Extension) — Push payload rendering only — no App Group or Keychain data accessed. No elevated data access identified; scoped strictly to expanded notification UI rendering. The Share Extension's unscoped read access to the full cached transaction-history file — rather than the single transaction the user selected for sharing — represents a data-isolation weakness between the main app and the extension and is tracked as IOS-017 (Medium). The Widget Extension's shared App Group cache is addressed separately as a data-at-rest exposure in Section 43 (IOS-010); the same Widget Extension's Keychain access-group scope is addressed under Keychain findings elsewhere in this report. The two Notification Extensions were not found to hold data access beyond their rendering role and are not associated with a finding in this section.

Finding ID	Severity	Title
IOS-017	MEDIUM	Improper App Extension Data Isolation

43 APP GROUPS

NovaSphere Pay defines a single shared App Group container, group.com.novasphere.pay.ios.shared, used to pass cached, at-a-glance data from the main app to its Home Screen Widget Extension without requiring the widget to launch the full app or hold its own network/session stack. TMG Security reviewed what is written to this shared container and how it is protected.

Illustrative Tests Performed

- Verify data written to a shared App Group container is limited to the minimum needed for the consuming extension's display purpose.
- Verify shared App Group data at rest is encrypted, or is otherwise not reversible to a sensitive underlying value, given that App Group membership is a sharing boundary rather than a confidentiality control.
- Verify the shared container is not treated as equivalent in protection to the main app's own private sandbox or Keychain-backed storage.

Illustrative Observations

The shared App Group container's widget_cache.plist stores the fictional account's cached balance and the last four digits of the linked card number in an unencrypted property list, readable by any other process or extension sharing the same App Group entitlement, and (on a jailbroken fictional test device) recoverable directly from the container's filesystem path with no additional barrier. Because App Group membership defines a sharing boundary rather than a confidentiality boundary, this plaintext storage extends the effective exposure of cached financial data beyond the main app's own sandbox. This is tracked as IOS-010 (High).

Finding ID	Severity	Title
IOS-010	HIGH	Insecure App Group Data Sharing

44 PASTEBOARD / CLIPBOARD SECURITY

TMG Security reviewed every code path in NovaSphere Pay that writes to UIPasteboard, focusing on whether sensitive values (account numbers, transfer references, one-time codes) are given an expiration, a locality restriction, or are avoided on the general pasteboard entirely.

Illustrative Tests Performed

- Verify sensitive values written to the pasteboard use UIPasteboard.setItems(_:options:) with an explicit expirationDate rather than the bare .string setter.

- Verify sensitive pasteboard items set localOnly to prevent propagation to Universal Clipboard on a paired Mac or other Continuity-enabled device.
- Verify a fictional test app polling UIPasteboard.general shortly after a sensitive copy action cannot recover the value indefinitely.
- Verify the app does not proactively read UIPasteboard.general on launch or foreground in a way that could ingest data the user did not intend to share with it.

Illustrative Observations

The "copy account number" convenience action writes the full fictional account number to UIPasteboard.general using the bare .string setter, with no expirationDate and no localOnly restriction. A fictional test app polling the general pasteboard after the copy action recovered the value in plaintext, and the same value was observed briefly available to Universal Clipboard on a paired fictional Mac. This is tracked as IOS-012 (Medium). No other pasteboard-writing code path in the app was found to carry a comparable exposure, and the app was not observed reading the general pasteboard proactively on launch or foreground.

Finding ID	Severity	Title
IOS-012	MEDIUM	Sensitive Data Exposure Through Pasteboard

45 DOCUMENT INTERACTION

TMG Security reviewed NovaSphere Pay's Info.plist file-sharing configuration and its use of UIDocumentInteractionController for the statement-export (PDF) feature, focusing on how broadly the app's on-device documents are exposed to the Files app, iTunes/Finder file sharing, and other apps able to open a shared document.

Illustrative Tests Performed

- Verify UIFileSharingEnabled and LSSupportsOpeningDocumentsInPlace, where set, are scoped to a narrow export-only directory rather than the app's entire Documents directory.
- Verify exported statement PDFs are removed from the shared file-sharing surface promptly after the user completes the share/export action.
- Verify UIDocumentInteractionController is presented only for files the user explicitly chose to export, with no broader document type or directory exposed as a side effect.

Illustrative Observations

NovaSphere Pay enables both UIFileSharingEnabled and LSSupportsOpeningDocumentsInPlace, which together expose the app's entire Documents directory — not a narrowly scoped export subfolder — to the Files app's "On My iPhone" browser and to a connected fictional Mac's Finder file-sharing panel. Cached exported statement PDFs remained visible through this surface with no observed automatic cleanup after the export action completed. This is tracked as IOS-016 (Medium).

Finding ID	Severity	Title
IOS-016	MEDIUM	Insecure Document Interaction / File Sharing Configuration

46 UIACTIVITY / SHARE SHEET SECURITY

TMG Security reviewed what data NovaSphere Pay attaches to the standard iOS share sheet (UIActivityViewController) across its share-triggering flows — transaction-receipt sharing, statement export, and referral/invite — to confirm only the data the user explicitly selected is exposed to the chosen share target.

Illustrative Tests Performed

- Verify each UIActivityViewController invocation is populated only with the specific item the user selected to share, not an unrelated broader data object.

- Verify excludedActivityTypes is used where appropriate to prevent a sensitive item from being routed to an inappropriate activity type (e.g., posting a financial receipt to a social-media activity).
- Verify no share-sheet flow inadvertently exposes clipboard-resident sensitive data (cross-reference Section 44) as a side effect of the Copy activity type.

Illustrative Observations

Across the transaction-receipt, statement-export, and referral share flows tested, each invocation of UIActivityViewController was populated only with the single item the user selected, with no unrelated account data attached as a side effect. The transaction-receipt share flow appropriately excludes sensitive activity types (e.g., posting) via excludedActivityTypes. The one caveat identified is that the share sheet's built-in Copy activity type inherits the same pasteboard exposure already tracked as IOS-012: an account number shared this way is copied to UIPasteboard.general without the expiration/locality hardening recommended in Section 44. No additional, independent share-sheet finding is raised in this section.

Finding ID	Severity	Title
IOS-012	MEDIUM	Sensitive Data Exposure Through Pasteboard

47 HANDOFF

TMG Security reviewed NovaSphere Pay's use of NSUserActivity-based Handoff, which allows a user to resume an in-progress task (e.g., viewing an account statement) on another of their fictional Continuity-enabled devices signed into the same Apple ID.

Illustrative Tests Performed

- Verify NSUserActivity objects published for Handoff do not embed sensitive data (account numbers, balances, tokens) directly in userInfo or in the activity's title/webpageURL.
- Verify a resumed Handoff activity re-establishes its data from an authenticated session on the receiving device rather than trusting data carried in the activity payload.
- Verify Handoff is not published for screens displaying transaction detail, card numbers, or other high-sensitivity content.

Illustrative Observations

NovaSphere Pay publishes NSUserActivity for only two lower-sensitivity navigation states (the account-overview landing screen and the support/help screen); neither activity's userInfo dictionary carries an account number, balance, or token. Resuming either activity on a fictional second device requires that device's own authenticated session before any data is displayed. No Handoff-related finding is raised; this section is reviewed as clean.

No findings raised against this fictional testing category beyond items cross-referenced elsewhere in this report.

48 SIRIKIT / SIRI SHORTCUTS

TMG Security enumerated the Siri Shortcuts actions NovaSphere Pay donates via NSUserActivity/ININtent donation and reviewed the Shortcuts app configuration surface (custom phrases, parameter prompts) for each donated action.

Illustrative Tests Performed

- Verify every donated Shortcuts action maps to a documented App Intent reviewed under Section 49.
- Verify no donated shortcut allows a custom phrase or parameter to substitute for an authentication step.
- Verify Shortcuts automation (e.g., a personal automation triggered by time or location) cannot chain a donated action into an unintended state-changing sequence.

Illustrative Observations

NovaSphere Pay donates two Shortcuts actions — "Check NovaSphere Pay Balance" and "Find Nearby NovaSphere ATM" — both of which map directly to the read-only App Intents reviewed in Section 49. Neither shortcut accepts a free-text parameter capable of altering its scope beyond the account already associated with the signed-in session, and no personal-automation chaining path was found capable of combining these actions into a state-changing sequence. No finding is raised in this section; the underlying App Intents surface is assessed in detail in Section 49.

No findings raised against this fictional testing category beyond items cross-referenced elsewhere in this report.

49 APP INTENTS / AI AGENT EXPOSURE

This section specifically evaluates NovaSphere Pay's App Intents surface against the risk of invocation not just by Siri and the Shortcuts app, but by any AI-agent-style caller capable of discovering and invoking a device's exposed App Intents (e.g., an on-device or third-party assistant enumerating available intents to act on the user's behalf) — a scope item of growing relevance for financial apps as agentic AI invocation of App Intents becomes more common on the platform.

Illustrative Tests Performed

- Enumerate every App Intent NovaSphere Pay exposes via the App Intents framework, independent of whether it is currently donated to Siri or Shortcuts.
- Verify no exposed App Intent can move funds, add or modify a payee, change an account limit, or otherwise mutate account state.
- Verify each exposed App Intent that returns account data (balance, nearby-ATM location) operates against the already-authenticated session and does not itself accept or elevate a credential.
- Verify no exposed App Intent's parameter schema allows an AI-agent-style caller to chain a read-only intent into a state-changing follow-on action without the app's own confirmation UI.

Illustrative Observations

TMG Security's enumeration of NovaSphere Pay's App Intents confirmed exactly two intents are exposed: a balance-check intent and a nearby-ATM-lookup intent, both strictly read-only and both scoped to the device's already-authenticated session. No funds-transfer, payee-management, or any other account-mutating intent is reachable through App Intents, Siri, Shortcuts, or (by extension) any AI-agent-style caller capable of discovering and invoking the app's declared intents. This is a positive control observation: limiting the voice/AI-agent-reachable surface to low-sensitivity, read-only actions is a sound design choice for a payments application, and TMG Security recommends NovaSphere continue reviewing any future App Intent against this same principle before exposing it. This observation is tracked as IOS-045 (Informational).

Finding ID	Severity	Title
IOS-045	INFORMATIONAL	Observation — Siri Shortcuts and App Intents Surface Is Appropriately Restricted

50 PUSH NOTIFICATION SECURITY

TMG Security reviewed the content and delivery configuration of NovaSphere Pay's APNs-delivered push notifications, focusing on transaction-alert notifications and what content is rendered on the lock screen before the device is unlocked.

Illustrative Tests Performed

- Verify sensitive fields (transaction amount, merchant name, account balance) are not present in the notification alert.body shown on a locked device by default.
- Verify a UNNotificationServiceExtension or content-redaction mechanism is used to reveal full detail only when appropriate (e.g., after unlock).

- Verify the device token registration and APNs payload delivery occur over an authenticated, encrypted channel with no sensitive data placed in unencrypted push metadata.
- Verify a user-facing setting exists (or is recommended) to control the level of detail shown on the lock screen.

Illustrative Observations

Transaction-alert push notifications display the full fictional transaction amount and merchant name directly in the lock-screen notification body by default, with no UNNotificationServiceExtension redaction applied and no per-user setting to reduce lock-screen detail. A fictional test transaction generated a lock-screen notification showing complete transaction detail while the device remained locked. This is tracked as IOS-019 (Medium). The underlying APNs delivery channel itself was confirmed to use standard encrypted, authenticated delivery, and device-token handling was not found to expose any additional weakness.

Finding ID	Severity	Title
IOS-019	MEDIUM	Sensitive Push Notification Data Exposure

51 BACKGROUND SNAPSHOT / SCREENSHOT SECURITY

TMG Security tested whether NovaSphere Pay installs a privacy overlay before the app is backgrounded, to prevent sensitive on-screen content from being captured into the iOS App Switcher's live preview and its on-disk snapshot cache.

Illustrative Tests Performed

- Verify every screen displaying account balance, card numbers, or transaction detail installs a privacy overlay in applicationDidEnterBackground/sceneDidEnterBackground.
- Verify the overlay is removed only after the app returns to the foreground and any local-authentication gate required for that screen has passed.
- Verify the on-disk App Switcher snapshot file does not retain unobscured sensitive content after the app returns to the foreground.

Illustrative Observations

The login and MFA screens correctly install a privacy overlay before backgrounding, but the same pattern was not extended to the account-overview and card-detail screens added in a later release. Backgrounding the app from the account-overview screen and opening the App Switcher showed the full fictional account balance and card number in the live, unobscured preview, which is also persisted to the on-disk snapshot cache. This is tracked as IOS-015 (Medium).

Finding ID	Severity	Title
IOS-015	MEDIUM	Sensitive Data Exposure Through Background App Snapshot

52 LOGGING & DEBUG OUTPUT

TMG Security reviewed NovaSphere Pay's use of os_log across the networking, authentication, and general diagnostic/telemetry modules, both for outright sensitive-value exposure and for consistency of privacy-annotation hygiene across lower-sensitivity call sites.

Illustrative Tests Performed

- Verify bearer tokens, MFA codes, and other credential material are never written to os_log in any build configuration distributed for QA or production use.
- Verify os_log calls that must log request/response detail apply {private} formatting by default rather than {public}.
- Verify lower-sensitivity diagnostic call sites (UI timing, cache-hit telemetry) still apply explicit privacy annotations rather than relying on default behavior.

- Verify no sensitive value logged during a live login flow is retrievable via the unified log (Console.app / log CLI) on a connected fictional Mac.

Illustrative Observations

The fictional QA/pre-production build configuration leaves verbose os_log(.debug, ...) statements active in the networking and authentication modules, logging full request headers — including the live bearer token — and the fictional MFA one-time code using {public} formatting. Streaming the unified log during a fictional login flow recovered both values in cleartext. This is tracked as IOS-008 (High). Separately, and at materially lower severity, several non-authentication diagnostic modules (UI navigation timing, cache-hit telemetry) use os_log calls without explicit {public}/{private} privacy annotations; the specific values observed at these call sites during testing were low-sensitivity, but the inconsistent annotation practice is itself a hygiene and defense-in-depth gap that could allow a future change to accidentally log sensitive data through an already-permissive code path. This is tracked separately as IOS-030 (Low).

Finding ID	Severity	Title
IOS-008	HIGH	Sensitive Data Exposure Through Application Logs
IOS-030	LOW	Logging Hygiene Weaknesses in Non-Sensitive Diagnostic Modules

53 PRIVACY PERMISSIONS

TMG Security reviewed NovaSphere Pay's full set of Info.plist usage-description keys and first-launch runtime permission requests against the app's currently shipping feature set, to identify any permission requested beyond what active functionality requires.

Illustrative Tests Performed

- Verify every requested dangerous/sensitive-category permission (Camera, Face ID, Contacts, Microphone, Calendars) maps to a currently active, shipping feature.
- Verify permissions are requested just-in-time at the point of feature use rather than batched at first launch.
- Verify usage-description keys and runtime requests left over from a removed or never-shipped feature have been fully cleaned up, not merely hidden from the UI.

Illustrative Observations

NovaSphere Pay requests Contacts and Microphone permissions at first launch alongside its core Camera (check deposit) and Face ID permissions, but neither Contacts nor Microphone access backs any currently shipping feature — both remain from a discontinued voice-memo-support feature and an abandoned contacts-based referral feature respectively. Requesting these permissions in a first-launch batch, with no corresponding active use, is tracked as IOS-022 (Medium). Static review additionally found the Info.plist still declares NSCameraUsageDescription-adjacent Calendars and a second, narrower set of stale usage-description keys with no reachable UI path requesting the underlying capability at all; this lower-severity, purely declarative residue is tracked separately as IOS-028 (Low). TMG Security recommends removing all unused usage-description keys and their associated runtime requests, and adopting just-in-time, feature-triggered permission prompts going forward.

Finding ID	Severity	Title
IOS-022	MEDIUM	Excessive Privacy Permission Requests Beyond Feature Requirements
IOS-028	LOW	Unused and Unnecessary Privacy-Sensitive Permission Declarations

54 THIRD-PARTY SDK REVIEW

TMG Security inventoried and reviewed every third-party SDK embedded in the NovaSpherePay.ipa build for its declared purpose, its data-access scope, and the security posture of its client-side integration. The full inventory appears below and in Appendix M.

SDK	Vendor (Fictional)	Purpose / Data Access	Governance Note	Finding Ref.

Illustrative Tests Performed

- Verify each embedded third-party SDK's data access is limited to its documented, feature-scoped purpose.
- Verify tracking or analytics identifiers persisted by a third-party SDK honor the platform's identity-reset boundaries (e.g., reset of identifierForVendor on full app deletion).
- Verify crash-reporting and diagnostic SDKs are configured and periodically reviewed to avoid capturing incidental PII in breadcrumbs or payloads.
- Verify the SDK inventory itself is complete, current, and independently reproducible from the compiled Mach-O binary's embedded frameworks.

Illustrative Observations

Of the ten fictional SDKs inventoried, the analytics SDK (AnalyticsKit) was found to persist its own tracking identifier in the Keychain rather than relying on identifierForVendor, so the identifier survives app deletion and reinstall in a way that undermines a user's expectation that reinstalling the app resets their analytics identity (IOS-023). No SDK was observed accessing data outside its documented purpose, and the mapping and OAuth account-linking SDKs' weaknesses are tracked individually elsewhere in this report (IOS-009, IOS-005) rather than re-scored here. TMG Security additionally notes that the inventory below was compiled specifically for this assessment rather than reflecting a continuously maintained internal record, and recommends NovaSphere formalize ongoing SDK inventory tracking (IOS-042). Separately, the crash-reporting SDK's default breadcrumb configuration has not been subject to a documented periodic review for incidental PII, which TMG Security flags as a defensive-baseline recommendation rather than a confirmed exposure (IOS-043).

Finding ID	Severity	Title
IOS-023	MEDIUM	Analytics SDK Persistent Identifier Reuse Across Reinstall
IOS-042	INFORMATIONAL	Recommendation — Formalize Third-Party SDK Inventory Tracking
IOS-043	INFORMATIONAL	Recommendation — Review Crash-Reporting Payloads for Incidental PII

55 DEPENDENCY SECURITY

TMG Security reviewed NovaSphere Pay's Swift Package Manager / CocoaPods dependency manifest (see Appendix M) for outdated or known-vulnerable third-party components, and for the governance process — or absence of one — managing dependency freshness and vulnerability scanning.

Illustrative Tests Performed

- Verify third-party dependencies are scanned for known vulnerabilities on a defined, recurring cadence.
- Verify no dependency carries a publicly disclosed, unpatched vulnerability applicable to its usage within NovaSphere Pay.
- Verify every third-party dependency has a documented owner, review cadence, and version-pinning policy.

Illustrative Observations

Two lower-usage fictional dependencies — an image-caching library (ImageCacheKit) and a deprecated supplementary analytics helper retained alongside AnalyticsKit — were found without a documented owner, review cadence, or version-pinning policy, unlike the app's core SDKs (IOS-031). No specific exploitable vulnerability was confirmed in either dependency's currently pinned version during this assessment's scope; the finding concerns governance process rather than a confirmed exploit path. TMG Security separately notes that dependency vulnerability scanning currently runs on an ad hoc basis rather than a fixed recurring schedule, and recommends NovaSphere move to a fixed, CI-integrated scanning cadence (IOS-046).

Finding ID	Severity	Title
IOS-031	LOW	Third-Party Dependency Governance Gaps
IOS-046	INFORMATIONAL	Recommendation — Increase Dependency-Scanning Cadence for Third-Party SDKs

56 JAILBREAK DETECTION

TMG Security tested NovaSphere Pay's jailbreak-detection routine on jailbroken fictional test devices, including devices configured with widely available jailbreak-hiding tooling, as the entry point into the fuller runtime-integrity assessment in Section 59.

Illustrative Tests Performed

- Verify the application detects common jailbreak indicators (Cydia-equivalent app presence, sandbox escape, suspicious dynamic libraries, restricted-path write access) at startup.
- Verify jailbreak detection cannot be trivially bypassed using a widely available jailbreak-hiding tweak.
- Verify a detected jailbroken environment triggers a defined defensive response consistent with the application's risk posture.

Illustrative Observations

NovaSphere Pay's jailbreak-detection routine checks only for the presence of /Applications/Cydia.app and whether fork() succeeds — both well-known, easily bypassed indicators. On a fictional jailbroken test device configured with a standard jailbreak-hiding tweak, neither check flagged the device as jailbroken, and the application proceeded normally with no elevated-risk signal raised to itself or its backend (IOS-018). This finding is rated Medium rather than higher because weak jailbreak detection is a defense-in-depth gap rather than a control that, on its own, grants unauthorized access to account data.

Finding ID	Severity	Title
IOS-018	MEDIUM	Weak Jailbreak / Runtime Integrity Protection

57 ANTI-DEBUGGING

TMG Security reviewed whether NovaSphere Pay implements anti-debugging measures — such as a ptrace(PT_DENY_ATTACH, ...) call or an equivalent runtime check — to detect and react to an attached fictional debugger, independent of jailbreak status.

Illustrative Tests Performed

- Verify the application calls ptrace(PT_DENY_ATTACH, ...) or an equivalent mechanism to resist a debugger attaching to the running process.
- Verify a detected debugger attachment triggers a defensive response (session termination or equivalent) rather than being silently permitted.
- Verify anti-debugging checks are not trivially bypassable by patching a single predictable code path.

Illustrative Observations

TMG Security's fictional dynamic-instrumentation testing was able to attach a debugger to the running NovaSphere Pay process on a jailbroken fictional test device without the application detecting or reacting to the attachment. No dedicated ptrace-deny (or equivalent) anti-debugging check was identified during static or dynamic review. TMG Security records this as part of the same broader weak-resilience posture already tracked as IOS-018 (Section 56) rather than as an independently scored finding, consistent with this report's treatment of MASVS-RESILIENCE controls as defense-in-depth: their absence does not itself constitute a directly exploitable vulnerability distinct from the jailbreak/runtime-integrity gap already captured.

No findings raised against this fictional testing category beyond items cross-referenced elsewhere in this report.

58 ANTI-TAMPERING

TMG Security reviewed NovaSphere Pay's runtime tamper-detection controls — code-signature self-verification and any use of a platform attestation service — for coverage and bypass resistance, extending the jailbreak-detection review in Section 56.

Illustrative Tests Performed

- Verify the application verifies its own code-signing configuration at runtime and reacts appropriately to an inconsistency.
- Verify the application uses a platform attestation service (e.g., DeviceCheck or App Attest) rather than relying solely on client-side self-checks.
- Verify tamper-detection checks are validated against a fictional jailbroken, instrumented test device rather than assumed effective from code review alone.

Illustrative Observations

NovaSphere Pay was found to perform no dedicated runtime tamper-detection check beyond the weak jailbreak-detection routine covered in Section 56 (IOS-018); no code-signature self-verification routine and no DeviceCheck/App Attest integration were identified during static or dynamic review. TMG Security records this gap as part of the same broader weak-resilience theme already tracked as IOS-018 rather than as a separate finding, and separately raises the absence of a platform attestation signal as a forward-looking recommendation (IOS-038, Section 59) rather than a confirmed exploitable weakness in its own right.

No findings raised against this fictional testing category beyond items cross-referenced elsewhere in this report.

59 RUNTIME INTEGRITY

TMG Security consolidated the jailbreak-detection (Section 56), anti-debugging (Section 57), and anti-tampering (Section 58) results into a single runtime-integrity assessment, and additionally reviewed whether NovaSphere Pay's device/session-assurance model relies on any server-verifiable platform attestation signal.

Illustrative Tests Performed

- Verify the application's client-side integrity checks, taken together, resist a fictional actively-instrumented jailbroken test environment.
- Verify device or session assurance relies on more than a client-generated value alone where a platform-attested alternative exists.
- Verify a detected loss of runtime integrity triggers a defensive response consistent with the application's other resilience checks.

Illustrative Observations

Taken together, the weak Cydia-path/fork() jailbreak check (Section 56) and the absence of any dedicated anti-debugging or anti-tampering routine (Sections 57-58) mean NovaSphere Pay's client-side integrity controls do not reliably detect a compromised, actively instrumented fictional device (IOS-018, not re-scored here). TMG Security also confirmed no use of Apple's DeviceCheck or App Attest frameworks for server-side device assurance; the application instead relies on a client-generated device-binding nonce, itself flagged elsewhere in this report as using a weaker PRNG than recommended (IOS-024). TMG Security recommends NovaSphere evaluate adopting DeviceCheck or App Attest to supplement or replace the current client-generated nonce approach with a platform-attested, server-verifiable signal (IOS-038).

Finding ID	Severity	Title
IOS-018	MEDIUM	Weak Jailbreak / Runtime Integrity Protection
IOS-038	INFORMATIONAL	Recommendation — Adopt DeviceCheck or App Attest for Server-Side Device Assurance

60 REVERSE ENGINEERING RESISTANCE

TMG Security assessed the decompiled fictional Mach-O binary's readability as a proxy for symbol and control-flow obfuscation coverage across NovaSphere Pay's sensitive business-logic modules.

Illustrative Tests Performed

- Verify class, struct, and method names are obfuscated in the release build across sensitive payments-logic modules.
- Verify control-flow or string-literal obfuscation is applied to security-relevant constants and logic beyond simple identifier naming.

Illustrative Observations

Static review of the Mach-O binary's symbol table found readable Swift type and method names throughout the application, including within payments-logic modules, with no symbol or control-flow obfuscation applied at build time. This meaningfully eased TMG Security's fictional static-analysis review of the transfer-confirmation and Keychain-access code paths referenced elsewhere in this report. TMG Security recommends NovaSphere evaluate applying obfuscation tooling to sensitive business-logic modules as a complement to — not a replacement for — the runtime resilience controls reviewed in Sections 56-59 (IOS-039).

Finding ID	Severity	Title
IOS-039	INFORMATIONAL	Recommendation — Evaluate Binary Obfuscation for Sensitive Business Logic

61 UPDATE ENFORCEMENT

TMG Security reviewed NovaSphere Pay's forced-update mechanism — the client-side check that compares the installed build against a fictional minimum-supported-version value returned by the backend — for its ability to reliably push users onto a build carrying a critical security patch.

Illustrative Tests Performed

- Verify the application checks a server-supplied minimum-supported-version value at startup and on resume.
- Verify a build below the minimum-supported version is blocked from continued use until updated, rather than only warned.
- Verify the enforcement check cannot be trivially bypassed (e.g., by operating fully offline) for security-relevant releases.

Illustrative Observations

NovaSphere Pay's forced-update mechanism was reviewed and found adequate for its intended purpose: the fictional test build correctly compared its version against a server-supplied minimum and presented a blocking update screen when set below that threshold, consistent with build 6402 (version 6.4.2) reporting itself correctly to the check. No finding is raised against this control area. TMG Security notes, as a related process observation rather than a technical gap in the enforcement mechanism itself, that the broader mobile release process was found to lack a documented internal security sign-off checkpoint gating when a build is deemed to carry a critical security patch in the first place (IOS-044, also referenced in Section 62); this is a governance recommendation about the release process surrounding update enforcement, not a defect in the enforcement mechanism tested here.

Finding ID	Severity	Title
IOS-044	INFORMATIONAL	Recommendation — Establish a Secure-SDLC Checkpoint for Mobile Releases

62 SECURE CONFIGURATION REVIEW

TMG Security reviewed NovaSphere Pay's build, CI/CD, and engineering-process configuration for the process-level controls that support (or fail to support) secure delivery of the client over time, complementing the technical findings detailed elsewhere in this report.

Illustrative Tests Performed

- Verify automated regression tests exist in CI covering previously identified security-relevant configuration (e.g., Keychain accessibility attributes, ATS policy).
- Verify an internal secure-coding guideline specific to iOS exists, covering Keychain usage, ATS configuration, and URL scheme/Universal Link handling.
- Verify build-time configuration values (API keys, environment settings) are sourced through a centralized secrets-management process rather than ad hoc inclusion.
- Verify a documented internal security checkpoint gates the mobile release process, independent of periodic external assessments such as this one.

Illustrative Observations

TMG Security's review of the fictional CI/CD pipeline found no automated regression tests specifically targeting previously identified security-relevant configuration, creating risk that a future refactor or dependency upgrade could silently reintroduce an already-remediated issue (IOS-033). Interviews with the fictional mobile engineering team further indicated no internal secure-coding guideline specific to iOS exists covering Keychain usage patterns, ATS configuration, or URL scheme/Universal Link handling, with consistent practice instead relying on individual engineer judgment and informal knowledge transfer (IOS-034). Beyond the confirmed hardcoded-key finding tracked elsewhere in this report (IOS-009), TMG Security found no centralized secrets-management tooling governing how build-time configuration values are sourced and injected via xcconfig files, and recommends NovaSphere formalize this process (IOS-040). Finally, interviews with release management found no formal, documented security sign-off checkpoint gating mobile releases; TMG Security recommends establishing a lightweight, documented secure-SDLC checkpoint as part of the standard release process (IOS-044, also referenced in Section 61 with respect to update-enforcement governance).

Finding ID	Severity	Title
IOS-033	LOW	Missing Automated Security Regression Test Coverage
IOS-034	LOW	Incomplete Secure-Coding Documentation for iOS Engineering
IOS-040	INFORMATIONAL	Recommendation — Formalize Secrets Management for Build-Time Configuration
IOS-044	INFORMATIONAL	Recommendation — Establish a Secure-SDLC Checkpoint for Mobile Releases

63 MOBILE API INTEGRATION SECURITY

Section 36 of this report detailed the specific broken object-level authorization defect in the funds-transfer confirmation endpoint (IOS-003). This section takes a broader, integration-level view of how NovaSphere Pay's iOS client handles authorization and session state across the mobile API surface as a whole, rather than repeating that endpoint-level detail.

Illustrative Tests Performed

- Verify every account-scoped mobile-API endpoint — not only the transfer-confirmation workflow — independently validates server-side that a referenced account identifier belongs to the authenticated session.
- Verify the bearer token used across all mobile-API calls is validated consistently by every endpoint, with no endpoint trusting client-supplied identity context instead of the token subject.
- Verify session and refresh-token lifecycle handling (issuance, refresh, revocation) is applied consistently across the full mobile API surface, not only at initial login.

Illustrative Observations

Viewed across the mobile API surface as a whole, NovaSphere Pay's client consistently attaches the bearer token to every authenticated request and relies on the server to authorize each call; the specific gap is not client-side but concentrated in how the transfer-confirmation endpoint validates the token's subject against the client-supplied sourceAccountId, detailed fully in Section 36 (IOS-003, not re-scored here). This integration-level review also surfaced the session-management angle covered separately in this report: because logout does not trigger server-side refresh-token revocation, a token captured

through any other exposure path remains usable against the broader mobile API surface — including the same account-scoped endpoints reviewed here — well past the point the user believes they have signed out (IOS-014). TMG Security recommends the account-ownership validation pattern already correctly applied on most endpoints be extended without exception to the transfer-confirmation workflow, and that server-side token revocation be integrated into the logout flow, so that both the authorization and session-lifecycle angles of the mobile API surface are addressed together rather than endpoint-by-endpoint.

Finding ID	Severity	Title
IOS-003	CRITICAL	Broken Authorization in Mobile API Workflow
IOS-014	HIGH	Missing Refresh Token Revocation After Logout

64 PRIVACY & DATA PROTECTION REVIEW

Sections 53 and 54 of this report documented specific privacy-relevant findings in isolation — excessive and unused permission declarations, and a third-party analytics SDK's persistent device identifier. This section takes a holistic view of NovaSphere Pay's privacy posture, synthesizing those individual observations into an overall assessment of how the fictional application collects, retains, and shares personal data relative to its stated functionality.

Illustrative Tests Performed

- Verify every requested privacy-sensitive permission (Contacts, Microphone, Camera, Calendars, Location) corresponds to an actively reachable feature.
- Verify third-party SDKs do not generate or persist device- or user-level identifiers that survive an app reinstall.
- Verify crash-reporting and diagnostic pipelines are periodically reviewed for incidentally captured personally identifiable information (PII).
- Verify the application's aggregate data-collection footprint is proportionate to its core payments functionality.

Illustrative Observations

Viewed together, NovaSphere Pay's privacy findings describe a consistent pattern rather than isolated defects: permissions and data-collection surfaces were added alongside specific features (branch-locator location access, pay-a-friend contact lookup, an early camera-based check-deposit concept, a calendar-reminder concept) and were not fully retired when those features were narrowed or shelved, leaving Microphone and Contacts access broader than current functionality requires (IOS-022) and Camera/Calendars usage-description keys present with no reachable UI path (IOS-028). Separately, the fictional analytics SDK's device-identifier behavior — persisting across reinstall via a Keychain-backed value rather than resetting with the application's own data (IOS-023) — represents a data-retention practice inconsistent with user expectations around uninstalling an app to remove its data, even though no crash-reporting PII leakage was confirmed during this assessment (IOS-043 remains a forward-looking recommendation rather than a confirmed exception). TMG Security recommends NovaSphere adopt a recurring privacy-surface review — ideally tied to each feature's lifecycle rather than performed only during periodic external assessments — covering permission necessity, third-party SDK data retention behavior, and crash/diagnostic payload content together, rather than as three separate concerns.

Finding ID	Severity	Title
IOS-022	MEDIUM	Excessive Privacy Permission Requests Beyond Feature Requirements
IOS-028	LOW	Unused and Unnecessary Privacy-Sensitive Permission Declarations
IOS-023	MEDIUM	Analytics SDK Persistent Identifier Reuse Across Reinstall
IOS-043	INFORMATIONAL	Recommendation — Review Crash-Reporting Payloads for Incidental PII

65 ATTACK CHAIN ANALYSIS

Individual findings are often most meaningful in combination. This section presents three fictional composite attack-chain scenarios, each illustrating how independently scored findings from this assessment could combine into a more significant illustrative impact if left unaddressed together — one rooted in a custom URL scheme, one rooted in a silently-actioned Universal Link, and one rooted in local Keychain/token exposure.

Attack Chain 1 — Custom URL Scheme to OAuth Token Theft

Fictional composite attack-chain scenario — not a real-world exploit chain.

An attacker registers a malicious fictional application on the same device that claims the same `novasphere://` custom URL scheme used by NovaSphere Pay's OAuth authentication redirect (IOS-005). When the legitimate app initiates its OAuth flow, iOS has no way to guarantee which app receives the redirect if more than one app has claimed the scheme, and the attacker's app can intercept the redirect callback carrying the authorization code or token. Because the scheme is a general-purpose custom scheme rather than a Universal Link bound to a verified associated domain, no domain-ownership check protects the handoff. The result is a captured fictional OAuth token that the attacker can use to impersonate the victim's authenticated session.

Attack Chain 2 — Universal Link Payee Injection to Cross-Account Fraud

Fictional composite attack-chain scenario — not a real-world exploit chain.

An attacker delivers a crafted `https://app.novasphere-example.com/link-payee` Universal Link to an authenticated fictional victim, for example embedded in a phishing message or malicious QR code. Because the Universal Link handler's `auto=1` fast path commits a new payee immediately with no in-app confirmation (IOS-004), the attacker-controlled payee is silently added to the victim's account the moment the link is opened. The attacker then leverages the broken object-level

authorization in the mobile API's transfer-confirmation workflow (IOS-003) to substitute the victim's account identifier into a transfer request authorized under a different fictional session, moving funds toward the newly-planted payee. The composite result — silent payee injection combined with a server-side authorization gap — produces a materially greater illustrative fraud impact than either finding presents in isolation.

Attack Chain 3 — Keychain Weak Access Control to Persistent Account Takeover

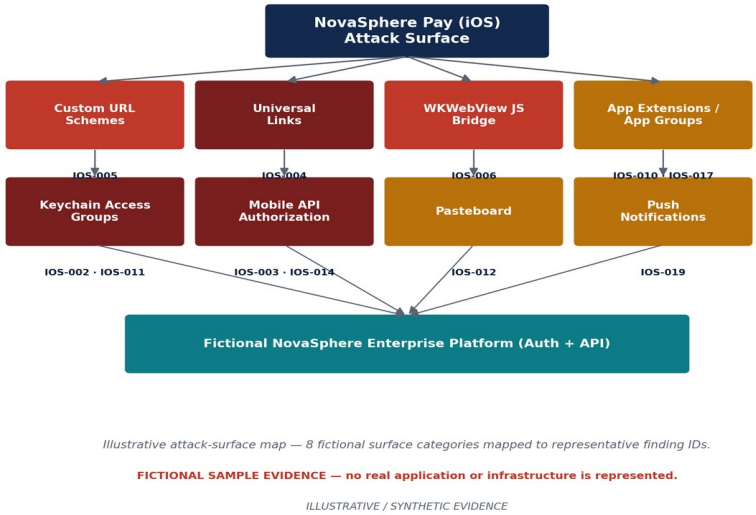
Fictional composite attack-chain scenario — not a real-world exploit chain.

On a fictional jailbroken or otherwise compromised test device, the biometric-vault Keychain item — stored with the least restrictive `kSecAttrAccessibleAlways` accessibility class and no `SecAccessControl` policy (IOS-002) — is extracted directly, exposing the material it protects. Separately, the application's authentication token is also found written in cleartext to an NSUserDefaults-backed property-list file rather than the Keychain (IOS-001), giving the attacker a second, independent path to the same session material. Because the refresh token is never revoked server-side when the user logs out (IOS-014), a token captured through either path remains valid well past the point the victim believes they have signed out. The attacker replays the still-valid token to re-establish the victim's authenticated session on an attacker-controlled device, achieving persistent fictional account access without needing to defeat biometric or passcode re-authentication at all.

THESE ARE FICTIONAL COMPOSITE ATTACK SCENARIOS. They do not describe, and must not be construed as, exploit chains against any real system.

66 SECURITY ARCHITECTURE OBSERVATIONS

Stepping back from individual findings, this section offers a high-level, illustrative view of NovaSphere Pay's overall iOS attack surface and the architectural themes running through this assessment's findings.



Illustrative attack-surface map — fictional entry points reachable on the NovaSphere Pay iOS client.

Recurring Architectural Themes

Theme	Illustrative Observation
Trust at IPC boundaries	Custom URL schemes and the Universal Link payee-linking path both trusted the incoming request more than their respective platform guarantees support (IOS-005, IOS-004).
Local storage inconsistency	Sensitive material was found in both under-protected Keychain configuration (IOS-002) and non-Keychain local storage (IOS-001, IOS-026) — no single, consistently-applied local-storage standard was evident across modules.
Server-side trust of client-supplied identity	The transfer-confirmation endpoint's authorization gap (IOS-003) reflects the same class of issue as several lower-severity API findings — trusting a client-supplied identifier over server-side validation of the authenticated session.
Session lifecycle gaps outlive individual findings	Session timeout (IOS-020) and refresh-token revocation (IOS-014) gaps mean that other exposure paths (IOS-001, IOS-002) remain exploitable for longer than they otherwise would.
Sharing surfaces added faster than isolation was hardened	App Groups (IOS-010), Keychain access groups (IOS-011), and App Extension data isolation (IOS-017) each independently under-scoped what an extension or widget can reach.

None of these themes individually rises to the severity of the Critical findings detailed in Section 70, but together they suggest an opportunity for NovaSphere's engineering organization to adopt a small number of platform-level standards — a single local-storage policy, a single IPC-trust validation pattern, and a single session/token lifecycle contract — that would prevent this class of finding from recurring across future features, rather than remediating each instance individually.

67 POSITIVE SECURITY CONTROLS

This fictional assessment is not intended to be entirely negative. TMG Security identified a number of security controls operating effectively across NovaSphere Pay's iOS client and its mobile API integration, summarized below as illustrative fictional observations.

Control Area	Illustrative Observation
Multi-Factor Account Recovery	The account-recovery flow consistently requires both an email OTP and an SMS OTP before permitting a credential reset, a sound control against recovery-flow-based account takeover (IOS-036).
TLS Enforcement (Primary Endpoints)	The primary API endpoints negotiate TLS 1.2/1.3 only, with strong cipher suites and no legacy downgrade path, aside from the isolated legacy-delegate issue tracked separately (IOS-037).
Restricted Siri / App Intents Surface	Only low-sensitivity, read-only App Intents (balance check, nearby-ATM lookup) are exposed to Siri, Shortcuts, and AI-agent-style invocation; no funds-movement intent is reachable through this surface (IOS-045).
CryptoKit Usage	Cryptographic operations reviewed during Section 26 correctly use Apple's CryptoKit framework rather than a custom or legacy cryptographic implementation, aside from the isolated weak-randomness gap tracked separately (IOS-024).
Code Signing & Provisioning	The current distribution certificate and provisioning profile were found validly configured and appropriately scoped, with the identified gap limited to lifecycle documentation rather than the current configuration (IOS-047).
Binary Hardening Defaults	Standard stack-canary and position-independent-executable (PIE) protections are present in the release Mach-O binary, with the identified gap limited to automated CI verification of this remaining true on every future build (IOS-035).
Object-Level Authorization (Partial)	Server-side account-ownership checks are correctly enforced on the majority of mobile-API endpoints tested — the confirmed gap is concentrated in the specific funds-transfer confirmation workflow detailed in Section 36 (IOS-003).

All observations above are fictional and were constructed for this illustrative sample assessment.

68 TESTING LIMITATIONS

The following limitations apply to this fictional sample assessment and should be read together with the Important Disclaimer presented in the report's front matter.

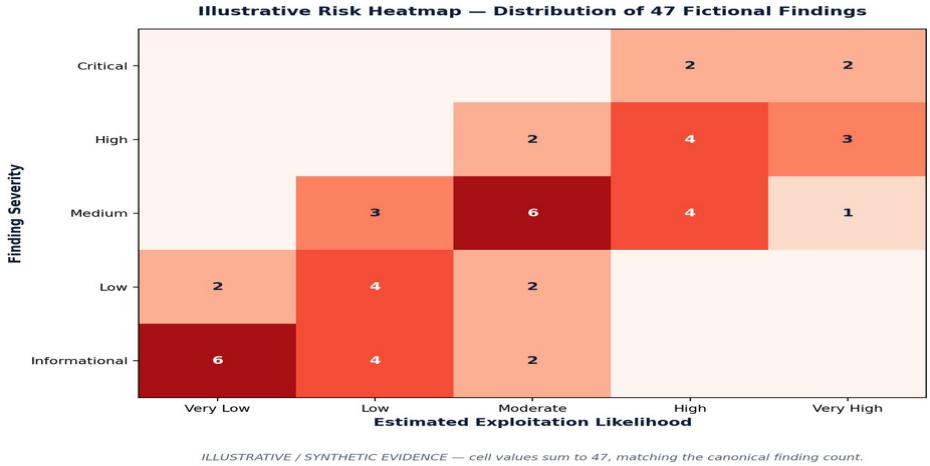
- NovaSphere Digital Technologies, Inc. and the NovaSphere Pay iOS application are fictional; no real organization or mobile application was assessed.
- No real credentials, API keys, signing certificates, or production systems were used or accessed.
- No real customer data was accessed at any point.
- No production environment or production backend was tested; only fictional test devices and a fictional controlled assessment environment are represented.
- No destructive testing was performed or simulated as executed, including against the fictional funds-transfer workflow.
- No denial-of-service testing was performed.
- No real Apple App Store distribution channel, code-signing infrastructure, or App Review process was tested or accessed.
- No social engineering was performed.
- No physical device theft or physical security testing was performed.
- No third-party SDK vendor's backend infrastructure was tested; SDK review was limited to the client-side integration observable within the fictional IPA.

- Testing was performed against fictional jailbroken and non-jailbroken test devices; results do not represent an exhaustive device- or iOS-version-fragmentation matrix.
- All evidence, screenshots, console/os_log output, otool/codesign output, and request/response examples in this report are synthetic.

This report describes a fictional, illustrative testing exercise only.

69 FINDINGS SUMMARY

This fictional assessment identified 47 illustrative findings across NovaSphere Pay's iOS client (bundle ID com.novasphere.pay.ios, version 6.4.2, build 6402) and its supporting mobile API integration, summarized below: 4 Critical, 9 High, 14 Medium, 8 Low, and 12 Informational. Full detail for every finding, including safe proof-of-concept evidence and OWASP MASVS/MASWE/MASTG traceability, is provided in Sections 70-74, organized by severity.



Illustrative risk heatmap — likelihood versus impact — ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE.

Findings at a Glance

Finding ID	Severity	Title	Status
IOS-001	CRITICAL	Sensitive Authentication Token Exposure Through Insecure Local Storage	Remediation In Progress
IOS-002	CRITICAL	Insecure Keychain Access Control Allows Unauthorized Credential Access	Remediation In Progress
IOS-003	CRITICAL	Broken Authorization in Mobile API Workflow	Remediation In Progress
IOS-004	CRITICAL	Insecure Universal Link / Deep-Link Flow Enables Sensitive Account Action	Remediation In Progress
IOS-005	HIGH	Insecure Custom URL Scheme Enables Authentication or Account Flow Hijacking	Remediation Planned
IOS-006	HIGH	WKWebView JavaScript Bridge Exposes Sensitive Native Functionality	Remediation Planned
IOS-007	HIGH	Weak Certificate Validation / Trust Evaluation Logic	Remediation Planned
IOS-008	HIGH	Sensitive Data Exposure Through Application Logs	Remediation Planned
IOS-009	HIGH	Hardcoded API Secret / Credential Material in iOS Binary	Remediation Planned
IOS-010	HIGH	Insecure App Group Data Sharing	Remediation Planned
IOS-011	HIGH	Keychain Access Group Misconfiguration	Remediation Planned
IOS-013	HIGH	Biometric Authentication Bypass / Improper LocalAuthentication Enforcement	Remediation Planned
IOS-014	HIGH	Missing Refresh Token Revocation After Logout	Remediation Planned
IOS-012	MEDIUM	Sensitive Data Exposure Through Pasteboard	Remediation Planned
IOS-015	MEDIUM	Sensitive Data Exposure Through Background App Snapshot	Remediation Planned
IOS-016	MEDIUM	Insecure Document Interaction / File Sharing Configuration	Remediation Planned
IOS-017	MEDIUM	Improper App Extension Data Isolation	Remediation Planned
IOS-018	MEDIUM	Weak Jailbreak / Runtime Integrity Protection	Remediation Planned
IOS-019	MEDIUM	Sensitive Push Notification Data Exposure	Remediation Planned
IOS-020	MEDIUM	Weak Session Timeout Enforcement	Remediation Planned

Finding ID	Severity	Title	Status
IOS-021	MEDIUM	Insecure ATS Exception Configuration Permits Weak TLS on Secondary Endpoint	Remediation Planned
IOS-022	MEDIUM	Excessive Privacy Permission Requests Beyond Feature Requirements	Remediation Planned
IOS-023	MEDIUM	Analytics SDK Persistent Identifier Reuse Across Reinstall	Remediation Planned
IOS-024	MEDIUM	Insecure Random Number Generation for a Security-Relevant Value	Remediation Planned
IOS-025	MEDIUM	Verbose Error Handling Reveals Internal Implementation Details	Remediation Planned
IOS-026	MEDIUM	Insecure Object Persistence of Sensitive Local Cache	Remediation Planned
IOS-027	MEDIUM	Missing App Transport Security Domain-Level Granularity	Remediation Planned
IOS-028	LOW	Unused and Unnecessary Privacy-Sensitive Permission Declarations	Remediation Planned
IOS-029	LOW	Use of Deprecated iOS APIs With Superseded Secure Alternatives	Remediation Planned
IOS-030	LOW	Logging Hygiene Weaknesses in Non-Sensitive Diagnostic Modules	Remediation Planned
IOS-031	LOW	Third-Party Dependency Governance Gaps	Remediation Planned
IOS-032	LOW	Attack-Surface Hardening Opportunity — Excessive Registered URL Scheme and Extension Count	Remediation Planned
IOS-033	LOW	Missing Automated Security Regression Test Coverage	Remediation Planned
IOS-034	LOW	Incomplete Secure-Coding Documentation for iOS Engineering	Remediation Planned
IOS-035	LOW	Missing CI Verification of Stack-Canary and PIE Binary Protections	Remediation Planned
IOS-036	INFORMATIONAL	Positive Observation — Multi-Factor Authentication Enforced on Account Recovery	Observation Only
IOS-037	INFORMATIONAL	Positive Observation — Strong TLS Configuration on Primary API Endpoints	Observation Only
IOS-038	INFORMATIONAL	Recommendation — Adopt DeviceCheck or App Attest for Server-Side Device Assurance	Observation Only
IOS-039	INFORMATIONAL	Recommendation — Evaluate Binary Obfuscation for Sensitive Business Logic	Observation Only
IOS-040	INFORMATIONAL	Recommendation — Formalize Secrets Management for Build-Time Configuration	Observation Only
IOS-041	INFORMATIONAL	Recommendation — Add Step-Up Biometric Re-Authentication for High-Value Transactions	Observation Only
IOS-042	INFORMATIONAL	Recommendation — Formalize Third-Party SDK Inventory Tracking	Observation Only
IOS-043	INFORMATIONAL	Recommendation — Review Crash-Reporting Payloads for Incidental PII	Observation Only
IOS-044	INFORMATIONAL	Recommendation — Establish a Secure-SDLC Checkpoint for Mobile Releases	Observation Only
IOS-045	INFORMATIONAL	Observation — Siri Shortcuts and App Intents Surface Is Appropriately Restricted	Observation Only
IOS-046	INFORMATIONAL	Recommendation — Increase Dependency-Scanning Cadence for Third-Party SDKs	Observation Only
IOS-047	INFORMATIONAL	Recommendation — Document Provisioning Profile and Signing Certificate Lifecycle Management	Observation Only

Finding Format

Each finding card in Sections 70-74 presents: Finding ID, Title, Severity, CVSS-style Score and Vector (where applicable), CWE reference, Affected Component, Affected iOS Component, Description, Business Context, Technical Impact, Business Impact, Attack Preconditions (where applicable), a Safe Proof of Concept summary, Evidence Reference, Expected Behavior, Observed (Fictional) Behavior, Root Cause, Recommendation, Management Response, Owner/Priority/Target Date/Status, a Retest Recommendation, and an OWASP MASVS/MASWE/MASTG traceability strip. Critical and High findings add a Technical Evidence Summary (Attack Preconditions, Observed Result, Security Impact, Business Impact, Evidence Collected, Retest Validation).

Safe Proof-of-Concept Standard

Every Proof of Concept in this report is illustrative and fictional. All evidence references the fictional iOS application bundle com.novasphere.pay.ios, build NovaSpherePay-6.4.2.ipa (version 6.4.2, build 6402), fictional jailbroken and non-jailbroken test devices, and the non-resolving placeholder domains api.novasphere-example.com and auth.novasphere-example.com. No PoC in this report was executed against a real system, and none is destructive, weaponized, or intended to enable real-world exploitation. Any synthetic screenshot or mock-tool evidence depicts no real application, device, or data and is marked ILLUSTRATIVE / SYNTHETIC EVIDENCE.

OWASP Reference Basis

Every MASVS control, MASWE weakness, and MASTG test identifier referenced in Sections 70-74 is drawn from current, publicly verifiable OWASP Mobile Application Security material (MASVS, MASTG v2.0.0, MASWE v1.0.0), using the iOS-specific MASTG-TEST identifier range (mas.owasp.org/MASTG/tests/ios/...). Where a precise verified identifier could not be confidently confirmed for a specific control concept, this report describes the concept in narrative terms rather than presenting an invented identifier. This report is an independent illustrative sample and carries no OWASP affiliation, review, or certification.

CVSS Scoring Note

All CVSS-style scores and vectors shown in this report — including the illustrative vector strings on each Critical and High finding — are sample estimations constructed for this fictional exercise. They are not derived from any real vulnerability, are not submitted to any CVSS authority, and should not be treated as authoritative scoring for a real system.

ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE — every request, response, Keychain dump, Console/os_log capture, decompiled/otool excerpt, screenshot, CVSS score/vector, and evidence reference in Sections 70-74 is synthetic.

70 CRITICAL FINDINGS

The following 4 fictional Critical-severity findings represent this fictional assessment's highest-priority illustrative results, each reproducible with a controlled, non-destructive fictional Proof of Concept and each carrying a direct path toward fictional account, credential, or funds compromise on NovaSphere Pay for iOS.

IOS-001 — Sensitive Authentication Token Exposure Through Insecure Local Storage	**CRITICAL**
CVSS-Style Score: 9.0 (Critical) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-312 — Cleartext Storage of Sensitive Information**
CVSS Vector (Illustrative): `CVSS:3.1/AV:P/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:N — Illustrative vector only`	
Affected Component	Local session/token persistence layer
Affected iOS Component	*NSUserDefaults suite `com.novasphere.pay.ios.session` (unencrypted plist, not Keychain)*
Description	NovaSphere Pay persists the OAuth 2.0 access token, refresh token, and a fictional device-binding secret in a standard NSUserDefaults plist rather than in the iOS Keychain. On a fictional jailbroken test device, or via a fictional unencrypted iTunes/Finder backup, the token material was recoverable directly from the app's container without any Keychain-backed cryptographic barrier.
Business Context	NovaSphere Pay's session and refresh tokens are the fictional bearer credentials for all authenticated payment, transfer, and account-management operations against the NovaSphere Enterprise Platform API. Their confidentiality is foundational to every other authorization control in the application.
Technical Impact	Any fictional process, unencrypted backup extraction, or forensic acquisition with access to the app's container (jailbroken device, unencrypted local backup, or full device-level compromise) can read the tokens in cleartext with no additional cryptographic barrier.
Business Impact	Full fictional account takeover is possible without the victim's password: a recovered token can be replayed against the NovaSphere API to perform payments, transfers, and profile changes as the fictional victim user until the token is revoked.
Attack Preconditions	Local or backup-level access to the fictional device's app container (e.g., a jailbroken fictional test device, or a fictional unencrypted local backup extracted via a desktop backup tool) while the victim's session is active.
Safe Proof of Concept (Summary)	*On a fictional jailbroken test device, reading the app's container Preferences plist via a fictional filesystem browser returned the session and refresh tokens in cleartext. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-001, EV-IOS-002 (fictional Preferences plist extraction, MASTG-TEST-0300 static/dynamic review notes)*
Expected Behavior	Sensitive tokens should be stored using the iOS Keychain (kSecClassGenericPassword) with kSecAttrAccessibleWhenUnlockedThisDeviceOnly or stricter, so extraction of the app container's plist files does not yield usable credential material.
Observed (Fictional) Behavior	The fictional access token, refresh token, and device-binding secret were all stored in a standard, unencrypted NSUserDefaults plist.
Root Cause	The fictional session-management module was implemented against UserDefaults for convenience during early development and was never migrated to the Keychain-backed pattern recommended for credential material.
Recommendation	Migrate all session/token persistence to the iOS Keychain with an appropriate accessibility class (kSecAttrAccessibleWhenUnlockedThisDeviceOnly); ensure tokens are never written to NSUserDefaults, application support files, or any location outside the Keychain; add static-analysis tooling to flag any future plaintext persistence of token-like values.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will migrate token storage to the Keychain ahead of the next scheduled release, tracked as the top remediation priority from this fictional assessment.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | Immediate — 0-30 Days | 10 October 2026 | **Remediation In Progress**
Retest Recommendation	*Retest by repeating the fictional Preferences plist extraction against the remediated build and confirming no token material is recoverable outside the Keychain.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-STORAGE-1 — "The app securely stores sensitive data."	MASWE-0001 — Sensitive Data Stored Unencrypted in Private Storage	MASTG-TEST-0300 — References to APIs for Storing Unencrypted Data in Private Storage

FULL EVIDENCE PANEL — ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE

```
Filesystem Extraction (Fictional Jailbroken Test Device) (Fictional / Synthetic)
$ ssh mobile@fictional-test-device-a
iPhone-Fictional:~ mobile$ cd /var/mobile/Containers/Data/Application/<FICTIONAL-GUID>/Library/Preferences
iPhone-Fictional:Preferences mobile$ cat com.novasphere.pay.ios.session.plist
```

Recovered Plist Contents (Fictional / Synthetic) (Fictional / Synthetic)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>access_token</key>
    <string>eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.FICTIONAL_PAYLOAD.sig</string>
    <key>refresh_token</key>
    <string>rt_novasphere_fictional_7c1e9b40aa2f</string>
    <key>device_binding_secret</key>
    <string>db_fictional_ios_51f0ac3d</string>
    <key>token_issued_at</key>
    <integer>1799999999</integer>
</dict>
</plist>
```

Expected: Access and refresh tokens should be stored only in the iOS Keychain with an appropriate kSecAttrAccessible protection class, unrecoverable without device unlock/Keychain access.

Observed: Both tokens and the device-binding secret were recoverable in cleartext directly from the app's NSUserDefaults-backed Preferences plist.

Impact: Full fictional session and refresh-token compromise, enabling fictional account takeover and continued token replay after logout if the refresh token is not separately revoked.

ILLUSTRATIVE / SYNTHETIC EVIDENCE
no real device, account, or data

ssh mobile@fictional-test-device-a

```
$ ssh mobile@fictional-test-device-a
iPhone-Fictional:~ mobile$ cd .../Data/Application/<FICTIONAL-GUID>/Library/Preferences
iPhone-Fictional:Preferences mobile$ cat com.novasphere.pay.ios.session.plist

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "...PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>access_token</key>
    <string>eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.FICTIONAL_PAYLOAD.sig</string>
    <key>refresh_token</key>
    <string>rt_novasphere_fictional_7c1e9b40aa2f</string>
    <key>device_binding_secret</key>
    <string>db_fictional_ios_51f0ac3d</string>
    <key>token_issued_at</key>
    <integer>1799999999</integer>
</dict>
</plist>

# Result: tokens recovered in CLEARTEXT — NSUserDefaults plist, no Keychain barrier
```

Finding IOS-001 — Sensitive Authentication Token Exposure Through Insecure Local Storage

Synthetic mock-tool evidence — ILLUSTRATIVE / SYNTHETIC EVIDENCE. No real application, device, or data is depicted.

TECHNICAL EVIDENCE SUMMARY

Attack Preconditions	Local or backup-level access to the fictional device's app container (e.g., a jailbroken fictional test device, or a fictional unencrypted local backup extracted via a desktop backup tool) while the victim's session is active.
Observed Result	The fictional access token, refresh token, and device-binding secret were all stored in a standard, unencrypted NSUserDefaults plist.
Security Impact	Any fictional process, unencrypted backup extraction, or forensic acquisition with access to the app's container (jailbroken device, unencrypted local backup, or full device-level compromise) can read the tokens in cleartext with no additional cryptographic barrier.
Business Impact	Full fictional account takeover is possible without the victim's password: a recovered token can be replayed against the NovaSphere API to perform payments, transfers, and profile changes as the fictional victim user until the token is revoked.
Evidence Collected	EV-IOS-001, EV-IOS-002 (fictional Preferences plist extraction, MASTG-TEST-0300 static/dynamic review notes)
Retest Validation	Retest by repeating the fictional Preferences plist extraction against the remediated build and confirming no token material is recoverable outside the Keychain.

IOS-002 — Insecure Keychain Access Control Allows Unauthorized Credential Access	**CRITICAL**

CVSS-Style Score: 8.7 (Critical) — Illustrative CVSS v3.1-inspired score	CWE: CWE-522 — Insufficiently Protected Credentials
CVSS Vector (Illustrative): `CVSS:3.1/AV:P/AC:L/PR:N/UI:N/S:C/C:H/I:L/A:N` — Illustrative vector only	
Affected Component	Keychain credential storage layer
Affected iOS Component	*Keychain item `com.novasphere.pay.ios.biometric-vault` (kSecAttrAccessibleAlways, no ACL)*
Description	A subset of Keychain items used to gate the fictional biometric-unlock feature are stored with kSecAttrAccessibleAlways and no SecAccessControl (no biometry/passcode binding), rather than a device-locked accessibility class such as kSecAttrAccessibleWhenUnlockedThisDeviceOnly combined with a biometry-bound SecAccessControl. On a fictional jailbroken device, these items were readable outside of any biometric or passcode gate, and were also included in fictional unencrypted local backups.
Business Context	The biometric-vault Keychain item wraps a fictional long-lived device-binding key used to re-establish an authenticated session without a full login flow — a highly sensitive credential given how directly it maps to session re-establishment.
Technical Impact	A fictional jailbreak-level compromise, or extraction from a fictional unencrypted device backup, can read the Keychain item's contents without triggering any biometric or device-passcode challenge, because the item's accessibility class and access-control policy do not require one.
Business Impact	An attacker with fictional physical or backup-level access could recover the device-binding key and use it to re-establish a fictional authenticated session, bypassing the intended biometric gate entirely.
Attack Preconditions	Fictional jailbreak-level filesystem access to the Keychain database, or possession of a fictional unencrypted local device backup containing the vulnerable Keychain item.
Safe Proof of Concept (Summary)	*A fictional Keychain dumper run on a jailbroken test device enumerated the `com.novasphere.pay.ios.biometric-vault` item and returned its plaintext value with no biometric prompt. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-003 (fictional Keychain dump), EV-IOS-004 (fictional backup extraction listing)*
Expected Behavior	Sensitive Keychain items backing biometric re-authentication should be stored with a device-locked accessibility class and a biometry-bound SecAccessControl, and excluded from unencrypted backups.
Observed (Fictional) Behavior	The biometric-vault Keychain item used the least restrictive accessibility class available (kSecAttrAccessibleAlways) with no access-control policy.
Root Cause	The fictional biometric-unlock feature was implemented by copying an example from early-stage internal documentation that predated the team's current Keychain accessibility-class standard, and was not revisited during a later security review.
Recommendation	Re-create the biometric-vault Keychain item using kSecAttrAccessibleWhenUnlockedThisDeviceOnly and a SecAccessControl requiring current biometry enrollment; audit all other Keychain items in the app for the same accessibility-class weakness; add a static-analysis rule flagging kSecAttrAccessibleAlways usage.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will re-provision the affected Keychain item with a biometry-bound access-control policy in the next release.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | Immediate — 0-30 Days | 10 October 2026 | **Remediation In Progress**
Retest Recommendation	*Retest by repeating the fictional Keychain dump against the remediated build and confirming the item cannot be read without a live biometric/passcode challenge.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-STORAGE-1 — "The app securely stores sensitive data."	MASWE-0016 — Cryptographic Key Access Not Restricted	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

FULL EVIDENCE PANEL — ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE

Fictional Keychain Dump (Jailbroken Test Device) (Fictional / Synthetic)

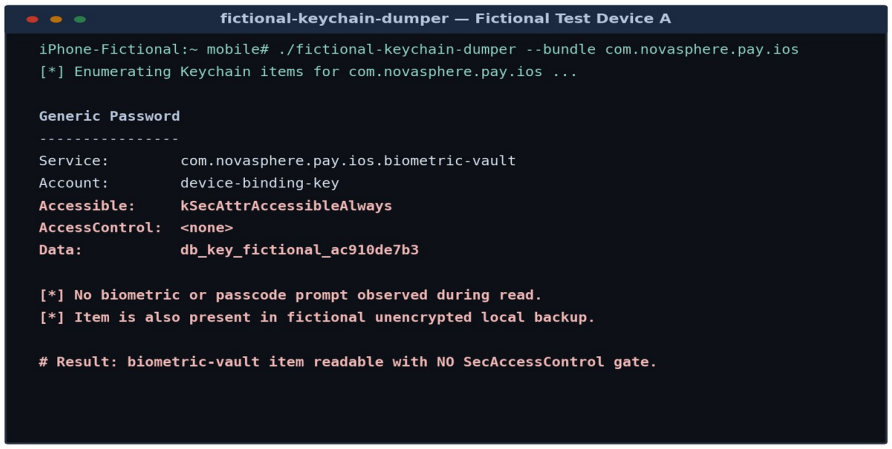
```
iPhone-Fictional:~ mobile# ./fictional-keychain-dumper --bundle com.novasphere.pay.ios
[*] Enumerating Keychain items for com.novasphere.pay.ios ...
Generic Password
---------------
Service: com.novasphere.pay.ios.biometric-vault
Account: device-binding-key
Accessible: kSecAttrAccessibleAlways
AccessControl: <none>
Data: db_key_fictional_ac910de7b3
```

Expected: The biometric-vault item should use kSecAttrAccessibleWhenUnlockedThisDeviceOnly together with a SecAccessControl requiring biometry (kSecAccessControlBiometryCurrentSet), so the item is unreadable without a live biometric or passcode challenge and is excluded from backups.

Observed: The item used kSecAttrAccessibleAlways with no SecAccessControl and was readable without any biometric prompt on a jailbroken fictional device.

Impact: Recovery of the device-binding key without biometric or passcode verification, enabling fictional session re-establishment outside the intended local-authentication gate.

ILLUSTRATIVE / SYNTHETIC EVIDENCE
no real device, account, or data



Finding IOS-002 — Insecure Keychain Access Control Allows Unauthorized Credential Access

Synthetic mock-tool evidence — ILLUSTRATIVE / SYNTHETIC EVIDENCE. No real application, device, or data is depicted.

TECHNICAL EVIDENCE SUMMARY

Attack Preconditions	Fictional jailbreak-level filesystem access to the Keychain database, or possession of a fictional unencrypted local device backup containing the vulnerable Keychain item.
Observed Result	The biometric-vault Keychain item used the least restrictive accessibility class available (kSecAttrAccessibleAlways) with no access-control policy.
Security Impact	A fictional jailbreak-level compromise, or extraction from a fictional unencrypted device backup, can read the Keychain item's contents without triggering any biometric or device-passcode challenge, because the item's accessibility class and access-control policy do not require one.
Business Impact	An attacker with fictional physical or backup-level access could recover the device-binding key and use it to re-establish a fictional authenticated session, bypassing the intended biometric gate entirely.
Evidence Collected	EV-IOS-003 (fictional Keychain dump), EV-IOS-004 (fictional backup extraction listing)
Retest Validation	Retest by repeating the fictional Keychain dump against the remediated build and confirming the item cannot be read without a live biometric/passcode challenge.

IOS-003 — Broken Authorization in Mobile API Workflow	**CRITICAL**
CVSS-Style Score: 9.1 (Critical) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-639 — Authorization Bypass Through User-Controlled Key**
CVSS Vector (Illustrative): `CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:N — Illustrative vector only`	
Affected Component	Mobile API — funds-transfer confirmation workflow
Affected iOS Component	*URLSession-based API client — POST /v3/mobile/transfers/confirm*
Description	The mobile-API transfer-confirmation endpoint accepts a client-supplied sourceAccountId parameter and processes the fictional transfer against that account without validating server-side that it belongs to the authenticated session. TMG Security's fictional testing confirmed that an authenticated iOS session for Test Account A could confirm a transfer sourced from Test Account B's fictional balance by substituting the account identifier in the request body.
Business Context	The funds-transfer workflow is the fictional application's most sensitive operation, moving money out of a NovaSphere Pay account on the user's behalf; object-level authorization on this endpoint is the single most important server-side control protecting customer funds.
Technical Impact	An authenticated fictional attacker can substitute another user's account identifier into the transfer-confirmation request and have the transfer processed against that account instead of their own, without any additional authorization check.
Business Impact	Direct fictional financial loss and fraud exposure: an attacker with any authenticated NovaSphere Pay session can move funds out of an arbitrary victim account, and the resulting fraud would initially appear to originate from the victim's own account.
Attack Preconditions	Any authenticated NovaSphere Pay session (a standard, non-privileged fictional test account) and knowledge or enumeration of a target account identifier.
Safe Proof of Concept (Summary)	*A fictional authenticated request substituting Test Account B's identifier into Test Account A's transfer-confirmation call was processed successfully against Test Account B's fictional balance. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-005 (fictional proxy capture, transfer-confirmation BOLA reproduction)*

Expected Behavior	Every account-scoped mobile-API request should independently validate server-side that the referenced account identifier belongs to the authenticated session before performing any state-changing action.
Observed (Fictional) Behavior	The transfer-confirmation endpoint trusted the client-supplied sourceAccountId without any server-side ownership check.
Root Cause	The transfer-confirmation endpoint was implemented to reuse a generic account-identifier parameter from an internal admin tool without adding the customer-facing object-level authorization check applied elsewhere in the platform's API layer.
Recommendation	Add a mandatory server-side check on every account-scoped mobile-API endpoint verifying that the referenced account identifier belongs to the authenticated session's own account set before processing; add automated authorization-regression tests covering this endpoint and its siblings; extend this check pattern to any other endpoint accepting a client-supplied account identifier.
Management Response	*NovaSphere engineering (fictional) has accepted this finding as a Critical-priority backend fix and has committed to deploying a hotfix ahead of the standard release cycle.*
Owner / Priority / Target / Status	**Backend/API Engineering Lead (Fictional Role)** | Immediate — 0-30 Days | 24 September 2026 | **Remediation In Progress**
Retest Recommendation	*Full retest required: repeat the cross-account substitution PoC against the remediated endpoint and confirm a 403 response; extend regression testing to all account-scoped mobile-API endpoints.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-AUTH-1 — "The app uses secure authentication and authorization protocols and follows the relevant best practices."	MASWE-0018 — Lack of Authentication or Authorization on App Components	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

FULL EVIDENCE PANEL — ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE

Fictional Proxy Capture — Request (Illustrative) (Fictional / Synthetic)

```
POST /v3/mobile/transfers/confirm HTTP/1.1
Host: api.novasphere-example.com
Authorization: Bearer [FICTIONAL-TOKEN-ACCOUNT-A]
Content-Type: application/json

{
  "transferId": "txn_fictional_88213",
  "sourceAccountId": "ACCT-559002",
  "destinationAccountId": "BEN-FICTIONAL-7741",
  "amount": 4200.00,
  "currency": "USD"
}
```

Fictional Proxy Capture — Response (Illustrative) (Fictional / Synthetic)

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "status": "CONFIRMED",
  "transferId": "txn_fictional_88213",
  "sourceAccountId": "ACCT-559002",
  "newBalance": 1180.44
}
```

Expected: The server should independently verify that sourceAccountId belongs to the account owned by the authenticated session (Bearer token subject) and reject the request with 403 Forbidden if not.

Observed: The transfer was confirmed and processed against ACCT-559002 (Test Account B) using a request authenticated as Test Account A's session, with no ownership validation performed server-side.

Impact: Complete object-level authorization bypass on the platform's highest-value operation, enabling arbitrary fictional cross-account funds transfer.

Synthetic mock-tool evidence — ILLUSTRATIVE / SYNTHETIC EVIDENCE. No real application, device, or data is depicted.

TECHNICAL EVIDENCE SUMMARY

Attack Preconditions	Any authenticated NovaSphere Pay session (a standard, non-privileged fictional test account) and knowledge or enumeration of a target account identifier.
Observed Result	The transfer-confirmation endpoint trusted the client-supplied sourceAccountId without any server-side ownership check.
Security Impact	An authenticated fictional attacker can substitute another user's account identifier into the transfer-confirmation request and have the transfer processed against that account instead of their own, without any additional authorization check.
Business Impact	Direct fictional financial loss and fraud exposure: an attacker with any authenticated NovaSphere Pay session can move funds out of an arbitrary victim account, and the resulting fraud would initially appear to originate from the victim's own account.
Evidence Collected	EV-IOS-005 (fictional proxy capture, transfer-confirmation BOLA reproduction)
Retest Validation	Full retest required: repeat the cross-account substitution PoC against the remediated endpoint and confirm a 403 response; extend regression testing to all account-scoped mobile-API endpoints.

IOS-004 — Insecure Universal Link / Deep-Link Flow Enables Sensitive Account Action	**CRITICAL**
CVSS-Style Score: 8.6 (Critical) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-940 — Improper Verification of Source of a Communication Channel**
CVSS Vector (Illustrative): `CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:H/A:N — Illustrative vector only`	
Affected Component	Universal Link handling — beneficiary-linking flow
Affected iOS Component	*NSUserActivity / `application(_:continue:restorationHandler:)` handler for `https://app.novasphere-example.com/link-payee`*
Description	NovaSphere Pay's Universal Link handler for `/link-payee` silently adds a new fictional payee to the authenticated user's account as soon as the link is opened, with no in-app confirmation step. A fictional attacker can deliver this link via phishing, and the moment an authenticated victim taps it, a new payee under attacker control is committed to their account without any additional user action.
Business Context	The payee-linking feature underlies NovaSphere Pay's peer-to-peer and bill-pay transfer flows; silently adding a payee removes the last user-facing checkpoint before a follow-on social-engineering-driven transfer.
Technical Impact	The NSUserActivity continuation handler for the `link-payee` path parses the payee parameters from the URL and calls the payee-creation API directly, with no confirmation UI and no re-authentication step, even though the action is state-changing and security-relevant.
Business Impact	An attacker can pre-position an attacker-controlled payee on a victim's fictional account via a single tapped link, then use social engineering to convince the victim to send a fictional transfer to that now-legitimate-looking, already-saved payee.
Attack Preconditions	The victim must be logged into NovaSphere Pay and must tap a fictional attacker-supplied Universal Link (e.g., delivered via a phishing message or malicious QR code).
Safe Proof of Concept (Summary)	*A fictional crafted Universal Link opened on an authenticated test device silently added a new payee with attacker-supplied bank details, with no confirmation screen shown. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-006 (fictional Universal Link trigger and console capture)*

Expected Behavior	Any Universal Link or custom-URL-scheme flow that results in a sensitive account change (adding a payee, changing a limit, modifying linked accounts) should require explicit in-app confirmation, not execute silently on link continuation.
Observed (Fictional) Behavior	The `link-payee` Universal Link path committed a new payee immediately, with an `auto=1` parameter explicitly bypassing the confirmation UI entirely.
Root Cause	The `auto=1` fast-path was added to streamline a fictional marketing-driven "one-tap payee setup" campaign and was never restricted to non-sensitive fields or gated behind confirmation.
Recommendation	Remove the silent auto-confirm path from the Universal Link handler; require explicit in-app confirmation, with the payee details clearly displayed, before any payee is committed via a link-originated flow; apply the same confirmation requirement to every other sensitive action reachable via Universal Link or custom URL scheme.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will remove the silent auto-confirm path in the next release, restoring mandatory in-app confirmation for all link-originated account changes.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | Immediate — 0-30 Days | 10 October 2026 | **Remediation In Progress**
Retest Recommendation	*Retest by repeating the fictional Universal Link PoC against the remediated build and confirming a confirmation screen is presented and no payee is committed without explicit user action.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-PLATFORM-1 — "The app uses IPC mechanisms securely."	MASWE-0029 — Insecure Deep Links	MASTG-TEST-0395 — Missing Input Validation in Universal Link Handlers

FULL EVIDENCE PANEL — ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE

Fictional Crafted Universal Link (Fictional / Synthetic)

```
https://app.novasphere-example.com/link-payee?name=Fictional%20Vendor%20LLC&accountRef=BEN-ATTACKER-0099&auto=1
```

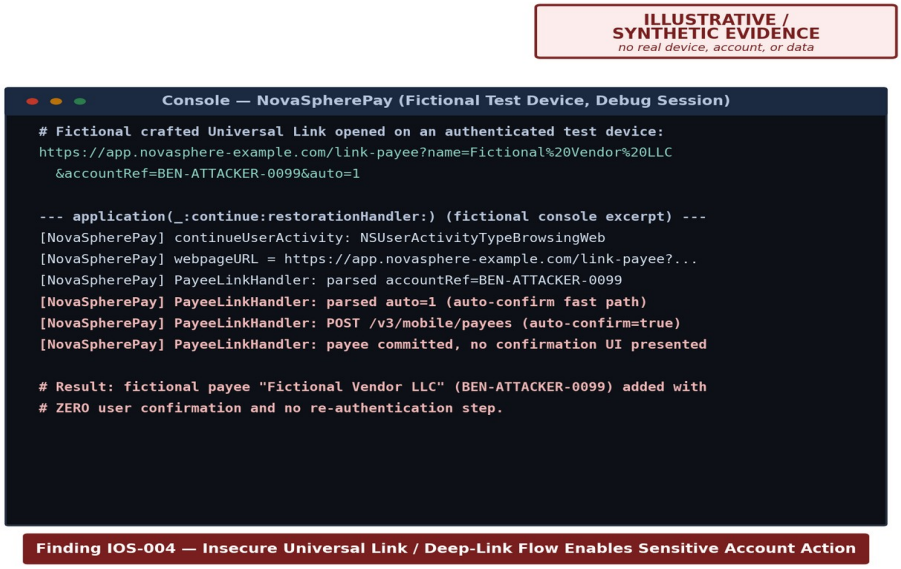
Console Output — NSUserActivity Continuation (Fictional Test Device) (Fictional / Synthetic)

```
[NovaSpherePay] continueUserActivity: https://app.novasphere-example.com/link-payee?name=Fictional%20Vendor%20LLC&accountRef=BEN-ATTACKER-0099&auto=1
[NovaSpherePay] PayeeLinkHandler: parsed accountRef=BEN-ATTACKER-0099
[NovaSpherePay] PayeeLinkHandler: POST /v3/mobile/payees (auto-confirm=true)
[NovaSpherePay] PayeeLinkHandler: payee committed, no UI presented
```

Expected: Opening a Universal Link that would add or modify account data should always present an in-app confirmation screen requiring explicit user action before the change is committed.

Observed: The payee was committed to the account immediately upon Universal Link continuation, with no confirmation screen shown to the user.

Impact: Silent, attacker-controlled modification of account state (a new payee) from a single tapped link, setting up a fictional social-engineering-driven funds transfer.



Synthetic mock-tool evidence — ILLUSTRATIVE / SYNTHETIC EVIDENCE. No real application, device, or data is depicted.

TECHNICAL EVIDENCE SUMMARY

Attack Preconditions	The victim must be logged into NovaSphere Pay and must tap a fictional attacker-supplied Universal Link (e.g., delivered via a phishing message or malicious QR code).
Observed Result	The `link-payee` Universal Link path committed a new payee immediately, with an `auto=1` parameter explicitly bypassing the confirmation UI entirely.
Security Impact	The NSUserActivity continuation handler for the `link-payee` path parses the payee parameters from the URL and calls the payee-creation API directly, with no confirmation UI and no re-authentication step, even though the action is state-changing and security-relevant.
Business Impact	An attacker can pre-position an attacker-controlled payee on a victim's fictional account via a single tapped link, then use social engineering to convince the victim to send a fictional transfer to that now-legitimate-looking, already-saved payee.
Evidence Collected	EV-IOS-006 (fictional Universal Link trigger and console capture)
Retest Validation	Retest by repeating the fictional Universal Link PoC against the remediated build and confirming a confirmation screen is presented and no payee is committed without explicit user action.

71 HIGH FINDINGS

The following 9 fictional High-severity findings should be prioritized immediately after the Critical findings in Section 70.

IOS-005 — Insecure Custom URL Scheme Enables Authentication or Account Flow Hijacking	HIGH
CVSS-Style Score: 7.8 (High) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-940 — Improper Verification of Source of a Communication Channel**
CVSS Vector (Illustrative): `CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:H/A:N — Illustrative vector only`	
Affected Component	Custom URL scheme handler — `novasphere://` OAuth redirect
Affected iOS Component	*`application(_:open:options:)` handler for the `novasphere://oauth-callback` path*
Description	NovaSphere Pay registers the `novasphere://` custom URL scheme, which iOS allows any installed app to also claim. The `novasphere://oauth-callback` path, used to complete a fictional third-party account-linking OAuth flow, accepts and processes an authorization code from any caller without verifying the code's origin, allowing a fictional malicious app to register the same scheme and intercept or inject a crafted callback.
Business Context	The OAuth account-linking flow is used to connect NovaSphere Pay to a fictional external bank-account aggregation partner; the callback URL carries a sensitive authorization code that, if hijacked, could be exchanged for access to the linked account data.
Technical Impact	Because custom URL schemes are not exclusively reserved to one app on iOS, a fictional malicious app registering the same `novasphere://` scheme could receive the OAuth callback instead of (or in addition to) NovaSphere Pay, or inject a spoofed callback directly.
Business Impact	A fictional attacker could hijack the account-linking OAuth flow to obtain an authorization code intended for NovaSphere Pay, potentially gaining unauthorized access to the linked external account data.
Attack Preconditions	A fictional malicious app installed on the same device, registered to claim the same `novasphere://` custom URL scheme, racing to handle the callback before or instead of the legitimate app.
Safe Proof of Concept (Summary)	*A fictional test harness app registered the `novasphere://` scheme and successfully received a simulated OAuth callback intended for NovaSphere Pay. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-007 (fictional URL scheme collision test)*
Expected Behavior	Security-sensitive redirect flows should use Universal Links instead of custom URL schemes, and should validate a per-request state/nonce value regardless of transport.
Observed (Fictional) Behavior	The OAuth callback used only the custom URL scheme with no observed state/nonce validation.
Root Cause	The account-linking feature was implemented before the team's Universal-Links-only policy for sensitive redirects was adopted, and was not retrofitted.
Recommendation	Migrate the OAuth callback to a Universal Link under the verified `app.novasphere-example.com` associated domain; add server-side state/nonce validation on the callback regardless of transport; deprecate the custom-scheme callback path entirely.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will migrate the OAuth callback to Universal Links in an upcoming release.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 31-60 Days | 15 November 2026 | **Remediation Planned**
Retest Recommendation	*Retest after migration to confirm the custom-scheme callback path is removed and the Universal Link callback validates state/nonce server-side.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-PLATFORM-1 — "The app uses IPC mechanisms securely."	MASWE-0029 — Insecure Deep Links	MASTG-TEST-0371 — Missing Source Validation in Custom URL Scheme Handlers

FULL EVIDENCE PANEL — ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE

```
Fictional Info.plist — CFBundleURLTypes (Test Harness App) (Fictional / Synthetic)
<key>CFBundleURLTypes</key>
<array>
  <dict>
    <key>CFBundleURLSchemes</key>
    <array><string>novasphere</string></array>
  </dict>
</array>
```

Expected: Sensitive OAuth callbacks should use Universal Links (which are cryptographically verified via the apple-app-site-association file) rather than a custom URL scheme, and should additionally validate a state/nonce parameter server-side.

Observed: The OAuth callback relied solely on the unreserved `novasphere://` custom URL scheme with no state/nonce validation observed in the fictional callback handler.

Impact: Potential interception or spoofing of the OAuth account-linking callback by a fictional co-installed malicious app claiming the same URL scheme.

TECHNICAL EVIDENCE SUMMARY

Attack Preconditions	A fictional malicious app installed on the same device, registered to claim the same `novasphere://` custom URL scheme, racing to handle the callback before or instead of the legitimate app.
Observed Result	The OAuth callback used only the custom URL scheme with no observed state/nonce validation.
Security Impact	Because custom URL schemes are not exclusively reserved to one app on iOS, a fictional malicious app registering the same `novasphere://` scheme could receive the OAuth callback instead of (or in addition to) NovaSphere Pay, or inject a spoofed callback directly.
Business Impact	A fictional attacker could hijack the account-linking OAuth flow to obtain an authorization code intended for NovaSphere Pay, potentially gaining unauthorized access to the linked external account data.
Evidence Collected	EV-IOS-007 (fictional URL scheme collision test)
Retest Validation	Retest after migration to confirm the custom-scheme callback path is removed and the Universal Link callback validates state/nonce server-side.

IOS-006 — WKWebView JavaScript Bridge Exposes Sensitive Native Functionality	**HIGH**
CVSS-Style Score: 7.6 (High) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-749 — Exposed Dangerous Method or Function**
CVSS Vector (Illustrative): `CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:L/A:N — Illustrative vector only`	
Affected Component	In-app support WKWebView — WKScriptMessageHandler bridge
Affected iOS Component	*`WKUserContentController.add(_:name:)` handler `novaBridge`, exposed to the support WKWebView*
Description	The support/help WKWebView exposes a `novaBridge` WKScriptMessageHandler with methods `getSessionToken` and `getDeviceIdentifier`, callable from any JavaScript running in the WebView, with no origin verification on the calling page. The WebView's navigation policy does not strictly allow-list first-party domains, widening the population of content that could reach the bridge.
Business Context	The support WebView is intended only to render first-party help content, but the bridge's live session-token exposure means any content able to execute JavaScript in that WebView context can retrieve the same token protected elsewhere in this report.
Technical Impact	JavaScript executing in the support WebView — whether first-party or reached through a navigation-policy gap — can call `window.webkit.messageHandlers.novaBridge.postMessage(...)` to retrieve the live session token and device identifier with no origin check performed by the handler.
Business Impact	Any script able to execute in the support WebView can silently exfiltrate the live session token, enabling fictional session hijacking without the user's knowledge.
Attack Preconditions	JavaScript execution in the support WKWebView context, reachable either through a compromised first-party support-content source or a gap in the WebView's navigation allow-list.
Safe Proof of Concept (Summary)	*A fictional test page loaded in the support WebView called the `novaBridge` handler directly and retrieved the live session token with no origin check. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-008 (fictional bridge exfiltration test)*
Expected Behavior	JavaScript bridges should expose the minimum necessary methods, verify the calling frame's origin before responding, and never return live session-token material to WebView-side script.
Observed (Fictional) Behavior	The `novaBridge` handler exposed a session-token getter with no origin check and no allow-list restriction on the WebView's navigation policy.
Root Cause	The bridge was implemented to simplify debugging of the support flow during development and was not scoped down before release.
Recommendation	Remove the session-token and device-identifier getters from the JavaScript bridge entirely; if native functionality must be exposed, verify `message.frameInfo.securityOrigin` against a strict first-party allow-list; restrict the support WebView's navigation policy to the same allow-list.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will remove the sensitive bridge methods and add origin verification in the next release.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 31-60 Days | 15 November 2026 | **Remediation Planned**
Retest Recommendation	*Retest by repeating the fictional bridge-call PoC against the remediated build and confirming the session-token getter is removed and origin verification is enforced.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-PLATFORM-2 — "The app uses WebViews securely."	MASWE-0034 — WebViews Allow Access to Local Resources with Untrusted Content	MASTG-TEST-0376 — References to Native Bridge APIs in WebViews

FULL EVIDENCE PANEL — ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE

Fictional Test Page — JavaScript (Synthetic) (Fictional / Synthetic)

```
window.webkit.messageHandlers.novaBridge.postMessage({
  action: "getSessionToken"
});
// Fictional handler response observed via console.log bridge echo:
// { token: "eyJhbGciOiJIUzI1NiIs...FICTIONAL", deviceId: "DEV-FICT-7731" }
```

Swift — WKScriptMessageHandler (Fictional Excerpt) (Fictional / Synthetic)

```
func userContentController(_ userContentController: WKUserContentController,
                           didReceive message: WKScriptMessage) {
    if message.name == "novaBridge" {
        // FIXME: no origin/frame verification performed before responding
        let token = SessionManager.shared.currentAccessToken
        respond(to: message, with: ["token": token, "deviceId": deviceIdentifier])
    }
}
```

Expected: The bridge handler should verify `message.frameInfo.securityOrigin` against a first-party allow-list before responding, and should never expose the live session token to WebView-side JavaScript at all.

Observed: The handler responded to any calling script with the live session token and device identifier, with no origin verification.

Impact: Session-token exfiltration from within the support WebView context, without requiring native code execution.

TECHNICAL EVIDENCE SUMMARY

Attack Preconditions	JavaScript execution in the support WKWebView context, reachable either through a compromised first-party support-content source or a gap in the WebView's navigation allow-list.
Observed Result	The `novaBridge` handler exposed a session-token getter with no origin check and no allow-list restriction on the WebView's navigation policy.
Security Impact	JavaScript executing in the support WebView — whether first-party or reached through a navigation-policy gap — can call `window.webkit.messageHandlers.novaBridge.postMessage(...)` to retrieve the live session token and device identifier with no origin check performed by the handler.
Business Impact	Any script able to execute in the support WebView can silently exfiltrate the live session token, enabling fictional session hijacking without the user's knowledge.
Evidence Collected	EV-IOS-008 (fictional bridge exfiltration test)
Retest Validation	Retest by repeating the fictional bridge-call PoC against the remediated build and confirming the session-token getter is removed and origin verification is enforced.

IOS-007 — Weak Certificate Validation / Trust Evaluation Logic	**HIGH**
CVSS-Style Score: 7.4 (High) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-295 — Improper Certificate Validation**
CVSS Vector (Illustrative): `CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:L/A:N — Illustrative vector only`	
Affected Component	Legacy statement-export network client — custom URLSessionDelegate
Affected iOS Component	*`URLSessionDelegate.urlSession(_:didReceive:completionHandler:)` custom trust evaluation (statement-export module)*
Description	A custom URLSessionDelegate implementation used only by the statement-export module (a legacy integration with a fictional third-party document backend) contains a certificate-chain validation defect: under a specific fictional proxy-interception condition involving an intermediate-certificate substitution, the custom logic accepts the connection where the platform default SecTrust evaluation would reject it.
Business Context	The statement-export module transmits fictional account statement data to a third-party document-generation backend; weakened trust evaluation on this channel could allow interception of exported financial statements.
Technical Impact	The custom trust-evaluation callback calls `completionHandler(.useCredential, URLCredential(trust: trust))` after an incomplete `SecTrustEvaluate` check that does not correctly propagate a chain-validation failure under the fictional intermediate-substitution scenario tested.
Business Impact	A fictional attacker positioned to intercept traffic on this legacy channel (e.g., via a rogue Wi-Fi access point) could, under this specific condition, present a substituted certificate and have it accepted, exposing exported statement data.
Attack Preconditions	Fictional network position to intercept the statement-export module's traffic (e.g., a rogue access point or ARP-spoofing

	position on a shared network) and a certificate crafted to match the specific validation gap.
Safe Proof of Concept (Summary)	*A fictional interception proxy configured with a substituted intermediate certificate successfully completed a TLS handshake against the statement-export module's custom trust evaluator. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-009 (fictional interception proxy capture against statement-export client)*
Expected Behavior	Every custom URLSessionDelegate trust-evaluation callback should enforce the result of SecTrustEvaluateWithError (or equivalent) as a hard gate before accepting a server credential.
Observed (Fictional) Behavior	The statement-export module's custom delegate accepted the server credential regardless of the trust-evaluation result.
Root Cause	The custom delegate was adapted from an older sample implementation that logged validation results for debugging and never enforced them as a pass/fail gate before the code reached production.
Recommendation	Fix the custom URLSessionDelegate to enforce SecTrustEvaluateWithError as a hard gate; where possible, remove the custom delegate entirely and rely on platform-default trust evaluation plus certificate pinning (Section 34); add automated interception-proxy testing to CI for all custom trust-evaluation code paths.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will correct the trust-evaluation logic in the next release, prioritized given the direct network-interception exposure.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 31-60 Days | 15 November 2026 | **Remediation Planned**
Retest Recommendation	*Retest by repeating the fictional certificate-substitution interception test against the remediated build and confirming the connection is rejected.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-NETWORK-2 — "The app performs identity pinning for all remote endpoints under the developer's control."	MASWE-0027 — Insecure Certificate Validation	MASTG-TEST-0396 — References to URLSessionDelegate Bypassing Certificate Validation

FULL EVIDENCE PANEL — ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE

Swift — Custom Trust Evaluation (Fictional Excerpt) (Fictional / Synthetic)

```
func urlSession(_ session: URLSession, didReceive challenge: URLAuthenticationChallenge,
                completionHandler: @escaping (URLSession.AuthChallengeDisposition, URLCredential?) ->
Void) {
    guard let trust = challenge.protectionSpace.serverTrust else {
        completionHandler(.performDefaultHandling, nil); return
    }
    // FIXME: result of SecTrustEvaluateWithError is logged but not enforced
    var error: CFError?
    _ = SecTrustEvaluateWithError(trust, &error)
    completionHandler(.useCredential, URLCredential(trust: trust))
}
```

Expected: The delegate should only call `completionHandler(.useCredential, ...)` when `SecTrustEvaluateWithError` returns true, and should call `.cancelAuthenticationChallenge` otherwise.

Observed: The delegate accepted the connection regardless of the SecTrustEvaluateWithError result, which was logged but not enforced as a gate.

Impact: TLS trust evaluation is effectively bypassed for the statement-export module's traffic, permitting interception under a crafted certificate-substitution scenario.

TECHNICAL EVIDENCE SUMMARY

Attack Preconditions	Fictional network position to intercept the statement-export module's traffic (e.g., a rogue access point or ARP-spoofing position on a shared network) and a certificate crafted to match the specific validation gap.
Observed Result	The statement-export module's custom delegate accepted the server credential regardless of the trust-evaluation result.
Security Impact	The custom trust-evaluation callback calls `completionHandler(.useCredential, URLCredential(trust: trust))` after an incomplete `SecTrustEvaluate` check that does not correctly propagate a chain-validation failure under the fictional intermediate-substitution scenario tested.
Business Impact	A fictional attacker positioned to intercept traffic on this legacy channel (e.g., via a rogue Wi-Fi access point) could, under this specific condition, present a substituted certificate and have it accepted, exposing exported statement data.
Evidence Collected	EV-IOS-009 (fictional interception proxy capture against statement-export client)
Retest Validation	Retest by repeating the fictional certificate-substitution interception test against the remediated build and confirming the connection is rejected.

IOS-008 — Sensitive Data Exposure Through Application Logs	**HIGH**

CVSS-Style Score: 7.1 (High) — Illustrative CVSS v3.1-inspired score	CWE: CWE-532 — Insertion of Sensitive Information into Log File
CVSS Vector (Illustrative): `CVSS:3.1/AV:P/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N — Illustrative vector only`	
Affected Component	Networking / authentication logging (os_log, pre-production build configuration)
Affected iOS Component	*`os_log(.debug, ...)` calls in `NetworkClient` and `AuthSessionManager`*
Description	A fictional QA/pre-production build configuration leaves verbose `os_log(.debug, ...)` statements active in the networking and authentication modules, writing full request headers — including the live bearer token — and the fictional MFA one-time code to the unified logging system, retrievable from the device via the Console app or `log` CLI on a connected fictional Mac.
Business Context	Verbose request/response logging is a common debugging aid, but in this fictional build configuration it was not stripped from the QA build channel used for pre-release validation, which is closer to production than a pure debug build.
Technical Impact	Any process able to read the unified log (a connected fictional Mac via Console.app/log CLI, or a device passcode-bypass condition) can recover the live bearer token and MFA code directly from log output.
Business Impact	Recovery of the live bearer token from logs enables the same fictional session-hijacking impact as other token-exposure findings in this report, via a log-access path rather than storage or Keychain access.
Attack Preconditions	Fictional physical or remote access sufficient to read the device's unified log stream (e.g., a connected Mac with the device paired, or a QA build distributed with verbose logging left enabled).
Safe Proof of Concept (Summary)	*Streaming the fictional test device's unified log via the `log` CLI during a login flow captured the live bearer token and MFA code in cleartext. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-010 (fictional unified-log capture during login)*
Expected Behavior	Sensitive values (tokens, MFA codes, credentials) should be redacted or entirely excluded from all logging, in every build configuration, using os_log's `{public}`/`{private}` privacy formatting correctly applied to mark such values private by default.
Observed (Fictional) Behavior	The QA build logged the full bearer token and MFA code using `{public}` formatting, making them visible in the unified log.
Root Cause	Debug logging added during initial development used `{public}` formatting for convenience and was not corrected before the QA build channel was established as the pre-release validation target.
Recommendation	Remove all sensitive-value logging from the networking and authentication modules; where request/response logging is retained for diagnostics, apply os_log `{private}` formatting to headers and body fields by default; add a static-analysis rule flagging `{public}` formatting applied to any variable sourced from token/credential storage.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will correct the logging configuration in the next QA and production build.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 31-60 Days | 15 November 2026 | Remediation Planned
Retest Recommendation	*Retest by repeating the fictional log-capture PoC against the remediated build and confirming no sensitive value appears in the unified log output.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-STORAGE-2 — "The app prevents leakage of sensitive data."	MASWE-0005 — Insertion of Sensitive Data into Logs	MASTG-TEST-0296 — Sensitive Data Exposure in Logs

FULL EVIDENCE PANEL — ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE

Fictional `log stream` Output (QA Build, Synthetic) (Fictional / Synthetic)

```
2026-09-08 10:14:02.118 NovaSpherePay[QA] [NetworkClient] Authorization: Bearer
eyJhbGciOiJIUzI1NiIs...FICTIONAL
2026-09-08 10:14:02.121 NovaSpherePay[QA] [AuthSessionManager] MFA code submitted: 482913
2026-09-08 10:14:02.130 NovaSpherePay[QA] [NetworkClient] Response 200: {"status":"authenticated"}
```

Expected: Bearer tokens, MFA codes, and other sensitive values should never be written to os_log at any log level, in any build configuration distributed for QA or production use.

Observed: The QA build configuration logged the full Authorization header and the fictional MFA code in cleartext at debug level.

Impact: Recovery of a live bearer token and MFA code via log access, without requiring Keychain or filesystem-level compromise.

TECHNICAL EVIDENCE SUMMARY

Attack Preconditions	Fictional physical or remote access sufficient to read the device's unified log stream (e.g., a connected Mac with the device paired, or a QA build distributed with verbose logging left enabled).
Observed Result	The QA build logged the full bearer token and MFA code using `{public}` formatting, making them visible in the unified log.
Security Impact	Any process able to read the unified log (a connected fictional Mac via Console.app/log CLI, or a device passcode-bypass condition) can recover the live bearer token and MFA code directly from log output.
Business Impact	Recovery of the live bearer token from logs enables the same fictional session-hijacking impact as other token-exposure findings in this report, via a log-access path rather than storage or Keychain access.

Evidence Collected	EV-IOS-010 (fictional unified-log capture during login)
Retest Validation	Retest by repeating the fictional log-capture PoC against the remediated build and confirming no sensitive value appears in the unified log output.

IOS-009 — Hardcoded API Secret / Credential Material in iOS Binary	**HIGH**
CVSS-Style Score: 7.3 (High) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-798 — Use of Hard-coded Credentials**
CVSS Vector (Illustrative): `CVSS:3.1/AV:P/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N — Illustrative vector only`	
Affected Component	Mapping/geolocation SDK integration layer
Affected iOS Component	*Mach-O `__TEXT.__cstring` section, `NovaSpherePay` binary*
Description	A fictional third-party mapping/geolocation SDK API key is hardcoded as a string literal compiled directly into the app's Mach-O binary, recoverable via a simple `strings` extraction with no decompilation required.
Business Context	The mapping SDK key gates access to a fictional paid geolocation service used for branch/ATM-locator functionality; exposure allows unauthorized fictional usage billed to NovaSphere's account and, depending on the fictional vendor's key scoping, potential access to usage analytics.
Technical Impact	The key is present in plaintext in the compiled binary's string table and requires no runtime instrumentation or decompilation to recover — a static `strings` pass against the extracted IPA payload is sufficient.
Business Impact	A fictional attacker could extract and reuse the mapping SDK key outside the app, incurring unauthorized usage costs and potentially exceeding the fictional vendor's rate limits, degrading the feature for legitimate users.
Attack Preconditions	Possession of the publicly distributed IPA (no device compromise or authentication required) and a standard binary string-extraction tool.
Safe Proof of Concept (Summary)	*Running `strings` against the extracted Mach-O binary recovered the mapping SDK API key in plaintext. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-011 (fictional strings extraction of Mach-O binary)*
Expected Behavior	Third-party SDK keys should not be compiled as plaintext string literals into the client binary; where a client-side key is unavoidable, it should be scoped and rate-limited server-side to the minimum necessary privilege.
Observed (Fictional) Behavior	The mapping SDK key was recoverable in plaintext via a static `strings` pass with no additional tooling.
Root Cause	The key was added directly to a Swift constants file during initial SDK integration, following the fictional vendor's quick-start documentation without a secrets-management review.
Recommendation	Move the mapping SDK key behind a server-side proxy endpoint or remote-config fetch gated by the app's authenticated session; rotate the currently exposed key; add a CI secret-scanning step to catch future hardcoded API keys before release.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will rotate the exposed key and move key provisioning behind an authenticated proxy in an upcoming release.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 31-60 Days | 15 November 2026 | **Remediation Planned**
Retest Recommendation	*Retest by repeating the fictional strings extraction against the remediated build and confirming no SDK key is present in the binary.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-STORAGE-1 — "The app securely stores sensitive data."	MASWE-0004 — Sensitive Data Hardcoded in the App Package	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

FULL EVIDENCE PANEL — ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE

Fictional `strings` Extraction (Fictional / Synthetic)

```
$ unzip -q NovaSpherePay-6.4.2.ipa -d extracted/
$ strings extracted/Payload/NovaSpherePay.app/NovaSpherePay | grep -i 'maps_api'
MAPS_API_KEY=fictional_maps_sdk_key_8a41cf209b
```

Expected: Third-party SDK API keys should be provisioned via a remote configuration fetch behind authentication, or restricted server-side to the app's known request signature, rather than compiled directly into the client binary.

Observed: The mapping SDK key was present as a plaintext string literal in the compiled binary.

Impact: Unauthorized extraction and reuse of the fictional mapping SDK key outside the application.

TECHNICAL EVIDENCE SUMMARY

Attack Preconditions	Possession of the publicly distributed IPA (no device compromise or authentication required) and a standard binary string-extraction tool.
Observed Result	The mapping SDK key was recoverable in plaintext via a static `strings` pass with no additional tooling.
Security Impact	The key is present in plaintext in the compiled binary's string table and requires no runtime instrumentation or decompilation to recover — a static `strings` pass against the extracted IPA payload is sufficient.
Business Impact	A fictional attacker could extract and reuse the mapping SDK key outside the app, incurring unauthorized usage costs and potentially exceeding the fictional vendor's rate limits, degrading the feature for legitimate users.
Evidence Collected	EV-IOS-011 (fictional strings extraction of Mach-O binary)
Retest Validation	Retest by repeating the fictional strings extraction against the remediated build and confirming no SDK key is present in the binary.

IOS-010 — Insecure App Group Data Sharing	**HIGH**
CVSS-Style Score: 7.0 (High) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-732 — Incorrect Permission Assignment for Critical Resource**
CVSS Vector (Illustrative): `CVSS:3.1/AV:P/AC:L/PR:N/UI:N/S:C/C:H/I:N/A:N — Illustrative vector only`	
Affected Component	Shared App Group container `group.com.novasphere.pay.ios.shared`
Affected iOS Component	*App Group container, consumed by the NovaSphere Pay Home Screen widget extension*
Description	The App Group container shared between NovaSphere Pay and its Home Screen widget extension stores a cached fictional account balance and the last four digits of the linked card number in an unencrypted plist, readable by any other extension or (on a fictional jailbroken device) any other process able to read the shared container path.
Business Context	The App Group is intended to let the widget display an at-a-glance fictional balance without launching the full app; the tradeoff is that any data written there is available to the whole App Group's trust boundary, not just the main app's sandbox.
Technical Impact	The shared container's plist is written without encryption, on the assumption that App Group isolation is a sufficient security boundary — it is a sharing boundary, not a confidentiality control, and does not protect the data from a fictional jailbreak-level compromise or a malicious future extension added to the same group.
Business Impact	A fictional attacker with jailbreak-level access, or a future compromised/malicious extension sharing the same App Group, could read the cached balance and partial card number without needing to compromise the main app's own sandbox.
Attack Preconditions	Fictional jailbreak-level filesystem access to the shared App Group container, or code execution within any other extension registered to the same App Group.
Safe Proof of Concept (Summary)	*A fictional filesystem read of the shared App Group container on a jailbroken test device returned the cached balance and partial card number in plaintext. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-012 (fictional App Group container extraction)*
Expected Behavior	Shared App Group data should be minimized to what the widget strictly needs to display and should not include reversible financial identifiers such as partial card numbers without additional protection.
Observed (Fictional) Behavior	The shared container stored the full cached balance and last-four card digits in plaintext.
Root Cause	The widget extension was built to read directly from a simple shared plist for development speed, and the data-minimization review applied to the main app's storage was not extended to the App Group container.
Recommendation	Reduce the App Group payload to the minimum display-necessary data; remove the card last-four digits from the shared cache entirely; if richer shared state is required, encrypt it with a Keychain-resident key not itself stored in the shared container.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will reduce the widget's shared-cache payload in the next release.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 31-60 Days | 15 November 2026 | **Remediation Planned**
Retest Recommendation	*Retest by repeating the fictional App Group extraction against the remediated build and confirming only minimized, non-sensitive display data is present.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-STORAGE-1 — "The app securely stores sensitive data."	MASWE-0001 — Sensitive Data Stored Unencrypted in Private Storage	MASTG-TEST-0388 — References to Sensitive Data Stored Unprotected in Shared App Group Containers

FULL EVIDENCE PANEL — ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE

```
Fictional Filesystem Access (Jailbroken Test Device) (Fictional / Synthetic)
iPhone-Fictional:~ mobile# find / -path '*group.com.novasphere.pay.ios.shared*' -name '*.plist'
/private/var/mobile/Containers/Shared/AppGroup/<FICTIONAL-GUID>/Library/Preferences/widget_cache.plist
iPhone-Fictional:~ mobile# plutil -p /private/var/.../widget_cache.plist
{
  "cachedBalance" => "$4,812.09"
```

```
  "cardLastFour" => "4417"
}
```

Expected: Data shared via an App Group container for widget display should be limited to the minimum necessary (e.g., a pre-formatted, non-reversible display string) or encrypted with a key not itself stored in the same container.

Observed: The shared container stored the cached balance and partial card number in a plaintext plist with no encryption.

Impact: Exposure of cached account balance and partial card number to any process sharing the App Group's trust boundary, beyond the main app's own sandbox.

TECHNICAL EVIDENCE SUMMARY

Attack Preconditions	Fictional jailbreak-level filesystem access to the shared App Group container, or code execution within any other extension registered to the same App Group.
Observed Result	The shared container stored the full cached balance and last-four card digits in plaintext.
Security Impact	The shared container's plist is written without encryption, on the assumption that App Group isolation is a sufficient security boundary — it is a sharing boundary, not a confidentiality control, and does not protect the data from a fictional jailbreak-level compromise or a malicious future extension added to the same group.
Business Impact	A fictional attacker with jailbreak-level access, or a future compromised/malicious extension sharing the same App Group, could read the cached balance and partial card number without needing to compromise the main app's own sandbox.
Evidence Collected	EV-IOS-012 (fictional App Group container extraction)
Retest Validation	Retest by repeating the fictional App Group extraction against the remediated build and confirming only minimized, non-sensitive display data is present.

IOS-011 — Keychain Access Group Misconfiguration	**HIGH**
CVSS-Style Score: 6.9 (High) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-284 — Improper Access Control**
CVSS Vector (Illustrative): `CVSS:3.1/AV:P/AC:L/PR:N/UI:N/S:C/C:H/I:N/A:N — Illustrative vector only`	
Affected Component	Keychain sharing configuration — app and widget extension
Affected iOS Component	*Entitlement `keychain-access-groups`: `$(AppIdentifierPrefix)com.novasphere.pay.ios.shared`*
Description	The Keychain access group shared between the main app and the Home Screen widget extension includes the same access group used to store the session refresh token, rather than a narrowly scoped access group limited to only the data the widget actually needs. Any code running inside the widget extension's process — including a future compromised or overly broad third-party widget dependency — can therefore read the refresh token from the shared Keychain access group.
Business Context	Widget extensions run in a more constrained but still code-executing context than the main app; sharing a security-critical Keychain access group with an extension unnecessarily widens the trust boundary around the refresh token.
Technical Impact	The refresh-token Keychain item and the widget's own cached-balance item both live in the same `keychain-access-groups` entitlement value, so any code with access to the widget extension's entitlements can query the shared Keychain access group and retrieve the refresh token.
Business Impact	A vulnerability or malicious dependency introduced into the widget extension in the future would have a direct path to the same refresh token protected by the Keychain-hardening work tracked elsewhere in this report, undermining that remediation.
Attack Preconditions	Code execution within the widget extension's process (e.g., via a future compromised third-party widget dependency, or fictional jailbreak-level Keychain enumeration scoped to the shared access group).
Safe Proof of Concept (Summary)	*A fictional test harness running inside the widget extension's entitlement context queried the shared Keychain access group and retrieved the refresh token alongside the widget's own cached data. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-013 (fictional Keychain access-group scope test)*
Expected Behavior	Keychain access groups should follow least privilege: each extension should share only a narrowly scoped access group containing the specific items it needs, never the same group used for high-sensitivity session material.
Observed (Fictional) Behavior	The refresh token and the widget's cache data were provisioned into the same shared Keychain access group.
Root Cause	A single shared access group was configured for simplicity when the widget extension was first added, without splitting session material into its own dedicated, main-app-only access group.
Recommendation	Create a separate Keychain access group scoped to the main app target only for the refresh token and other high-sensitivity session material; limit the shared App-Group-visible access group to the widget's own non-sensitive cache data.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will split the Keychain access groups in the next release.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 31-60 Days | 15 November 2026 | **Remediation Planned**

Retest Recommendation	*Retest by repeating the fictional Keychain access-group query from the widget extension context and confirming the refresh token is no longer reachable.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-PLATFORM-1 — "The app uses IPC mechanisms securely."	MASWE-0016 — Cryptographic Key Access Not Restricted	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

FULL EVIDENCE PANEL — ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE

Fictional Entitlements Excerpt — Widget Extension (Fictional / Synthetic)

```
<key>keychain-access-groups</key>
<array>
  <string>$(AppIdentifierPrefix)com.novasphere.pay.ios.shared</string>
</array>
```

Fictional Keychain Query (Widget Extension Context) (Fictional / Synthetic)

```
let query: [String: Any] = [
  kSecClass as String: kSecClassGenericPassword,
  kSecAttrAccessGroup as String: "com.novasphere.pay.ios.shared",
  kSecMatchLimit as String: kSecMatchLimitAll,
  kSecReturnAttributes as String: true
]
// Fictional result included both widget-cache and refresh_token items
```

Expected: The refresh token should live in an access group scoped only to the main app target; the widget extension should share a separate, narrowly scoped access group containing only its own display-cache data.

Observed: Both the refresh token and the widget's cache data shared the same Keychain access group.

Impact: Unnecessary widening of the refresh token's trust boundary to include the widget extension's process context.

TECHNICAL EVIDENCE SUMMARY

Attack Preconditions	Code execution within the widget extension's process (e.g., via a future compromised third-party widget dependency, or fictional jailbreak-level Keychain enumeration scoped to the shared access group).
Observed Result	The refresh token and the widget's cache data were provisioned into the same shared Keychain access group.
Security Impact	The refresh-token Keychain item and the widget's own cached-balance item both live in the same `keychain-access-groups` entitlement value, so any code with access to the widget extension's entitlements can query the shared Keychain access group and retrieve the refresh token.
Business Impact	A vulnerability or malicious dependency introduced into the widget extension in the future would have a direct path to the same refresh token protected by the Keychain-hardening work tracked elsewhere in this report, undermining that remediation.
Evidence Collected	EV-IOS-013 (fictional Keychain access-group scope test)
Retest Validation	Retest by repeating the fictional Keychain access-group query from the widget extension context and confirming the refresh token is no longer reachable.

IOS-013 — Biometric Authentication Bypass / Improper LocalAuthentication Enforcement	**HIGH**
CVSS-Style Score: 7.2 (High) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-287 — Improper Authentication**
CVSS Vector (Illustrative): `CVSS:3.1/AV:P/AC:L/PR:N/UI:N/S:C/C:H/I:L/A:N — Illustrative vector only`	
Affected Component	Biometric app-unlock gate
Affected iOS Component	*`LAContext.evaluatePolicy(_:localizedReason:reply:)` completion-handler result, checked but not cryptographically bound*
Description	NovaSphere Pay's Face ID app-unlock gate checks the boolean `success` result returned by `LAContext.evaluatePolicy`'s completion handler to decide whether to show the authenticated app state, but does not bind that result to a Keychain item protected by a matching SecAccessControl. On a fictional jailbroken test device, hooking the completion handler to force `success = true` bypassed the Face ID gate entirely.
Business Context	The app-unlock gate is the primary local-authentication checkpoint presented every time the app returns to the foreground; a bypassable implementation undermines the entire local-authentication control regardless of how strong the device's biometric enrollment is.
Technical Impact	Because the unlock decision is a plain boolean check in application code rather than a decision cryptographically tied to a

	successful Keychain operation gated by SecAccessControl, a fictional runtime-instrumentation hook that forces the completion handler's success parameter bypasses the entire gate.
Business Impact	An attacker with fictional jailbreak-level or instrumentation-level access to an unlocked device can bypass the Face ID gate without providing any valid biometric or passcode input, gaining full access to the authenticated app state.
Attack Preconditions	Fictional jailbreak-level or runtime-instrumentation-level access to the device (e.g., an instrumentation toolkit hooking the LAContext completion handler) with the device otherwise unlocked at the OS level.
Safe Proof of Concept (Summary)	*A fictional instrumentation script hooked `LAContext.evaluatePolicy`'s completion handler and forced a success result, bypassing the Face ID unlock gate without providing biometric input. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-014 (fictional instrumentation bypass of LAContext completion handler)*
Expected Behavior	Local-authentication decisions that gate sensitive app state should be anchored to a Keychain operation protected by a biometry-bound SecAccessControl, not a standalone boolean check in application code.
Observed (Fictional) Behavior	The unlock gate used only the LAContext completion handler's boolean result with no dependent Keychain operation.
Root Cause	The app-unlock feature was implemented using the simplest LocalAuthentication API pattern (a bare policy evaluation) rather than the recommended pattern of gating a Keychain item read behind SecAccessControl.
Recommendation	Rework the unlock gate to require successfully reading a Keychain item protected by kSecAccessControlBiometryCurrentSet, so the unlock decision is cryptographically anchored rather than a bare boolean; apply the same pattern to any other LocalAuthentication usage in the app.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will rework the unlock gate to use a Keychain-anchored biometric check in the next release.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 31-60 Days | 15 November 2026 | **Remediation Planned**
Retest Recommendation	*Retest by repeating the fictional instrumentation bypass against the remediated build and confirming a forced completion-handler result no longer grants access.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-AUTH-2 — "The app performs local authentication securely according to the platform best practices."	MASWE-0020 — Local Authentication Can Be Bypassed	MASTG-TEST-0266 — References to APIs for Event-Bound Biometric Authentication

FULL EVIDENCE PANEL — ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE

Fictional Instrumentation Console Output (Fictional / Synthetic)

```
[*] Hooking -[LAContext evaluatePolicy:localizedReason:reply:]
[*] Intercepted call, reply block located at 0x1042a3f10
[*] Forcing reply(YES, nil)
[*] NovaSpherePay: AppUnlockGate — biometric success=true, presenting authenticated state
[*] Bypass confirmed — no biometric input provided
```

Expected: The app-unlock decision should be tied to successfully unwrapping a Keychain item protected by a SecAccessControl requiring current biometry (kSecAccessControlBiometryCurrentSet), so a forced completion-handler result alone cannot substitute for a real biometric match.

Observed: The unlock gate relied solely on the completion handler's boolean result, with no dependent Keychain operation to cryptographically anchor the decision.

Impact: Complete bypass of the Face ID app-unlock gate under fictional jailbreak/instrumentation-level access, without any valid biometric input.

TECHNICAL EVIDENCE SUMMARY

Attack Preconditions	Fictional jailbreak-level or runtime-instrumentation-level access to the device (e.g., an instrumentation toolkit hooking the LAContext completion handler) with the device otherwise unlocked at the OS level.
Observed Result	The unlock gate used only the LAContext completion handler's boolean result with no dependent Keychain operation.
Security Impact	Because the unlock decision is a plain boolean check in application code rather than a decision cryptographically tied to a successful Keychain operation gated by SecAccessControl, a fictional runtime-instrumentation hook that forces the completion handler's success parameter bypasses the entire gate.
Business Impact	An attacker with fictional jailbreak-level or instrumentation-level access to an unlocked device can bypass the Face ID gate without providing any valid biometric or passcode input, gaining full access to the authenticated app state.
Evidence Collected	EV-IOS-014 (fictional instrumentation bypass of LAContext completion handler)
Retest Validation	Retest by repeating the fictional instrumentation bypass against the remediated build and confirming a forced completion-handler result no longer grants access.

IOS-014 — Missing Refresh Token Revocation After Logout	**HIGH**

CVSS-Style Score: 6.8 (High) — Illustrative CVSS v3.1-inspired score	CWE: CWE-613 — Insufficient Session Expiration
CVSS Vector (Illustrative): `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N — Illustrative vector only`	
Affected Component	Session lifecycle — logout flow
Affected iOS Component	*`SessionManager.logout()` — clears local Keychain item, no server-side revocation call*
Description	When a user logs out of NovaSphere Pay, the client removes the local Keychain-resident refresh token but does not call a server-side revocation endpoint. TMG Security's fictional testing confirmed that a refresh token captured prior to logout remained valid and could be replayed to obtain a new access token well after the user believed they had signed out.
Business Context	This finding directly extends the practical impact of any token-exposure finding elsewhere in this report (e.g., IOS-001, IOS-002): even where a token is captured, the victim's own logout action does not close the exposure window.
Technical Impact	The logout flow only clears client-side state; the fictional NovaSphere Enterprise Platform API does not receive or process any explicit token-revocation call, so a previously captured refresh token remains valid for its full original lifetime.
Business Impact	A refresh token captured through any other exposure path in this report remains usable even after the victim logs out, meaning "logging out" does not actually terminate an attacker's access if a token was already compromised.
Attack Preconditions	Prior capture of a valid refresh token (via any of this report's storage/Keychain findings), followed by the victim performing a normal logout.
Safe Proof of Concept (Summary)	*A fictional refresh token captured before logout was successfully replayed to obtain a new access token after the victim's fictional logout action. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-015 (fictional post-logout token-replay test)*
Expected Behavior	Logout should trigger an explicit server-side revocation call that invalidates the refresh token immediately, independent of client-side Keychain cleanup.
Observed (Fictional) Behavior	Logout cleared only the local Keychain item; the refresh token remained valid server-side.
Root Cause	The logout flow was implemented as a client-only operation during initial development, and a corresponding server-side revocation endpoint was never integrated into the logout call sequence.
Recommendation	Add a server-side token-revocation call to the logout flow, invalidating the refresh token immediately; ensure the revocation call is retried/queued if it fails while offline, so logout reliably revokes access once connectivity is restored.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will integrate server-side revocation into the logout flow in the next release.*
Owner / Priority / Target / Status	**Backend/API Engineering Lead (Fictional Role)** | 31-60 Days | 15 November 2026 | **Remediation Planned**
Retest Recommendation	*Retest by repeating the fictional post-logout token-replay test against the remediated flow and confirming the refresh token is rejected after logout.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-AUTH-1 — "The app uses secure authentication and authorization protocols and follows the relevant best practices."	MASWE-0024 — Sensitive Data Accessible After Session Termination	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

FULL EVIDENCE PANEL — ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE

```
Fictional Proxy Capture — Token Refresh After Logout (Illustrative) (Fictional / Synthetic)
POST /v3/mobile/auth/refresh HTTP/1.1
Host: auth.novasphere-example.com
Content-Type: application/json

{ "refreshToken": "rt_novasphere_fictional_7c1e9b40aa2f" }

HTTP/1.1 200 OK
{ "accessToken": "eyJhbGci...NEW_FICTIONAL", "expiresIn": 900 }
```

Expected: Logout should call a server-side revocation endpoint that immediately invalidates the refresh token, so a captured token cannot be used to obtain new access tokens after the user signs out.

Observed: The refresh token remained valid and issued a new access token after the victim's logout action.

Impact: Extended exposure window for any previously captured refresh token, undermining the user's own logout action as a mitigation.

TECHNICAL EVIDENCE SUMMARY

Attack Preconditions	Prior capture of a valid refresh token (via any of this report's storage/Keychain findings), followed by the victim performing a normal logout.
Observed Result	Logout cleared only the local Keychain item; the refresh token remained valid server-side.
Security Impact	The logout flow only clears client-side state; the fictional NovaSphere Enterprise Platform API does not receive or process any explicit token-revocation call, so a previously captured refresh token remains valid for its full original lifetime.

Business Impact	A refresh token captured through any other exposure path in this report remains usable even after the victim logs out, meaning "logging out" does not actually terminate an attacker's access if a token was already compromised.
Evidence Collected	EV-IOS-015 (fictional post-logout token-replay test)
Retest Validation	Retest by repeating the fictional post-logout token-replay test against the remediated flow and confirming the refresh token is rejected after logout.

72 MEDIUM FINDINGS

The following 14 fictional Medium-severity findings reflect defense-in-depth and platform-hardening gaps recommended for remediation within the 61-90 day window of the illustrative roadmap (Section 76).

IOS-012 — Sensitive Data Exposure Through Pasteboard	MEDIUM
CVSS-Style Score: 5.4 (Medium) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-200 — Exposure of Sensitive Information to an Unauthorized Actor**
Affected Component	"Copy account number" convenience action
Affected iOS Component	*`UIPasteboard.general.string` write, no `expirationDate` or `localOnly` option set*
Description	The "copy account number" action writes the full fictional account number to `UIPasteboard.general` with no expiration and no `localOnly` restriction, leaving it readable by any other app polling the general pasteboard shortly after the copy action, and (on supported OS versions) briefly visible to Universal Clipboard on a fictional paired Mac.
Business Context	The copy-account-number convenience feature is used when sharing account details with a third party (e.g., pasting into a banking app or email); its exposure window should be minimized given the sensitivity of the value.
Technical Impact	No `expirationDate` is set on the pasteboard item, so the value remains readable indefinitely until overwritten by another copy action, and no `localOnly` flag is set, allowing propagation to Universal Clipboard.
Business Impact	Any other fictional app polling the general pasteboard shortly after the copy action can read the account number without the user's awareness.
Safe Proof of Concept (Summary)	*A fictional test app polling `UIPasteboard.general.string` after the copy action recovered the account number in plaintext, with no expiration applied. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-016 (fictional pasteboard polling test)*
Expected Behavior	Copying a sensitive value should set an `expirationDate` (e.g., 60 seconds) and `localOnly: true` on the pasteboard item, using `UIPasteboard.setItems` with options rather than the bare `.string` setter.
Observed (Fictional) Behavior	The value was written using the bare `.string` setter with no expiration or locality restriction.
Root Cause	The convenience-copy feature used the simplest UIPasteboard API during implementation, and the security-hardened `setItems(_:options:)` API was not adopted.
Recommendation	Switch to `UIPasteboard.setItems(_:options:)` with `.expirationDate` set to a short interval and `.localOnly` set to true for any sensitive value copied to the pasteboard.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will add pasteboard expiration in a near-term release.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 61-90 Days | 15 December 2026 | **Remediation Planned**
Retest Recommendation	*Retest by repeating the fictional pasteboard polling test and confirming the value expires and is not propagated to Universal Clipboard.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-PLATFORM-3 — "The app uses the user interface securely."	MASWE-0036 — Unnecessary Exposure of Sensitive Data via the User Interface	MASTG-TEST-0276 — Use of the iOS General Pasteboard

IOS-015 — Sensitive Data Exposure Through Background App Snapshot	MEDIUM
CVSS-Style Score: 5.2 (Medium) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-200 — Exposure of Sensitive Information to an Unauthorized Actor**
Affected Component	Account overview and card-detail screens
Affected iOS Component	*No privacy overlay set in `applicationDidEnterBackground(_:)` / `sceneDidEnterBackground(_:)`*
Description	The account-balance and card-detail screens do not install a privacy overlay before the app is backgrounded, so the fictional account balance and full card number render in full in the iOS App Switcher snapshot and in the on-disk background snapshot image cached by the system.
Business Context	The App Switcher snapshot and its on-disk cache are accessible to anyone with brief physical access to an unlocked device, or via a fictional jailbreak-level read of the snapshot cache directory.
Technical Impact	No `UIView` overlay (e.g., a blurred brand splash) is inserted before backgrounding, so `UIApplication` captures the live, unobscured screen content into both the App Switcher and its persisted snapshot file.
Business Impact	Sensitive account and card details are exposed to anyone glancing at the App Switcher, or to a fictional jailbreak-level actor reading the cached snapshot file from disk.

Safe Proof of Concept (Summary)	*Backgrounding the app from the account-overview screen and opening the App Switcher showed the full fictional balance and card number in the live preview. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-017 (fictional App Switcher snapshot capture)*
Expected Behavior	Sensitive screens should install a privacy overlay (blurred view or brand splash) in `applicationDidEnterBackground`/`sceneDidEnterBackground` and remove it only after the app returns to the foreground and any local-authentication gate has passed.
Observed (Fictional) Behavior	No overlay was installed; the account-balance and card-detail screens were captured unobscured.
Root Cause	The privacy-overlay pattern was implemented for the login and MFA screens only, and was not extended to the account-balance and card-detail screens added in a later release.
Recommendation	Extend the existing privacy-overlay pattern to every screen displaying account balance, card numbers, or transaction detail; verify coverage with a dedicated backgrounding test for each sensitive screen.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will extend the privacy overlay to the affected screens.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 61-90 Days | 15 December 2026 | **Remediation Planned**
Retest Recommendation	*Retest by repeating the fictional backgrounding capture against the remediated build and confirming the overlay is shown on all sensitive screens.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-PLATFORM-3 — "The app uses the user interface securely."	MASWE-0038 — Insufficient Protection of Sensitive Data from Screenshots or Screen Recordings	MASTG-TEST-0290 — Runtime Verification of Sensitive Content Exposure in Screenshots During App Backgrounding

IOS-016 — Insecure Document Interaction / File Sharing Configuration	**MEDIUM**
CVSS-Style Score: 5.0 (Medium) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-668 — Exposure of Resource to Wrong Sphere**
Affected Component	Statement PDF export — UIDocumentInteractionController / Files app integration
Affected iOS Component	*`UIFileSharingEnabled` and `LSSupportsOpeningDocumentsInPlace` both set `true` in Info.plist*
Description	NovaSphere Pay enables both `UIFileSharingEnabled` and `LSSupportsOpeningDocumentsInPlace`, exposing the app's entire on-device Documents directory — including cached fictional exported statement PDFs — to the Files app and to iTunes/Finder file sharing, rather than exposing only a narrowly scoped export folder.
Business Context	Exported statement PDFs contain fictional transaction history and account details; broader-than-necessary file-sharing exposure widens the population of apps and desktop tools that can read them.
Technical Impact	Any exported statement PDF cached in the app's Documents directory is visible in the Files app's "On My iPhone" browser and via a connected fictional Mac's Finder file-sharing panel, for as long as the file remains cached.
Business Impact	A fictional attacker with brief physical access to an unlocked device, or access to a connected fictional Mac, could browse and extract cached statement PDFs through the Files app or Finder.
Safe Proof of Concept (Summary)	*Connecting a fictional test device to a fictional Mac and browsing file sharing for NovaSphere Pay showed cached statement PDFs in the shared Documents folder. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-018 (fictional Files app / Finder file-sharing browse)*
Expected Behavior	Only a narrowly scoped, purpose-built export subfolder should be exposed via file sharing, and cached statement PDFs should be deleted immediately after the user completes the share/export action.
Observed (Fictional) Behavior	The entire Documents directory was exposed via file sharing, and exported PDFs remained cached with no automatic cleanup observed.
Root Cause	File sharing was enabled broadly to support an early debugging workflow and was never scoped down or paired with cache cleanup before release.
Recommendation	Disable `UIFileSharingEnabled` for the Documents directory as a whole, or move exported PDFs to a `.novasphere-export` subfolder excluded from file sharing by default; delete cached export files immediately after the user completes the share action.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will scope file sharing to a dedicated export folder with automatic cleanup.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 61-90 Days | 15 December 2026 | **Remediation Planned**
Retest Recommendation	*Retest by repeating the fictional Files app/Finder browse against the remediated build and confirming only the scoped export folder is visible, with files cleaned up promptly.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-PLATFORM-1 — "The app uses IPC mechanisms securely."	MASWE-0002 — Sensitive Data Stored Unencrypted Outside of Private Storage (closest verified concept — primary concern is uncontrolled file-sharing exposure, not the absence of encryption)	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

IOS-017 — Improper App Extension Data Isolation	MEDIUM
CVSS-Style Score: 5.1 (Medium) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-668 — Exposure of Resource to Wrong Sphere**
Affected Component	Share-sheet action extension
Affected iOS Component	*`NovaSpherePay Share Extension` target — reads main app's cached transaction-history file directly from the App Group container*
Description	The share-sheet action extension (used to let a user quickly share a transaction receipt) reads the main app's full cached transaction-history file directly from the shared App Group container, rather than being handed only the single transaction the user selected, giving the extension unnecessarily broad access to unrelated transaction data.
Business Context	App extensions run with their own (typically more constrained) memory and entitlement context, but still execute third-party-reviewable code; the principle of least privilege applies to what data an extension can read, not just what it can do.
Technical Impact	The extension's data-access code opens the same cached transaction-history file the main app uses internally, rather than receiving a narrowly scoped payload (e.g., a single transaction's data) via the extension context passed at invocation.
Business Impact	A future vulnerability or malicious dependency in the share extension would have direct read access to the user's full cached transaction history, not just the single transaction being shared.
Safe Proof of Concept (Summary)	*Static review of the share-extension target's data-access code confirmed it reads the full shared transaction-history cache file rather than a scoped single-transaction payload. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-019 (fictional static review of share-extension data access)*
Expected Behavior	App extensions should receive only the minimum data needed for their specific task, passed explicitly via the extension context, rather than being given broad read access to a shared cache file.
Observed (Fictional) Behavior	The share extension reads the same full transaction-history cache file used internally by the main app.
Root Cause	The extension was implemented to reuse the main app's existing data-access layer for convenience, rather than introducing a scoped data-transfer object for the extension's specific use case.
Recommendation	Refactor the share extension to receive only the selected transaction's data via the extension item's user info dictionary, removing its direct read access to the shared transaction-history cache file.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will scope the share extension's data access in an upcoming release.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 61-90 Days | 15 December 2026 | **Remediation Planned**
Retest Recommendation	*Retest by reviewing the remediated share-extension code and confirming it no longer reads the full transaction-history cache file.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-PLATFORM-1 — "The app uses IPC mechanisms securely."	MASWE-0001 — Sensitive Data Stored Unencrypted in Private Storage (closest verified concept — primary concern is excessive data scope handed to the extension, not the absence of encryption)	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

IOS-018 — Weak Jailbreak / Runtime Integrity Protection	MEDIUM
CVSS-Style Score: 5.3 (Medium) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-693 — Protection Mechanism Failure**
Affected Component	Jailbreak-detection routine
Affected iOS Component	*Startup check for `/Applications/Cydia.app` and `fork()` availability only*
Description	NovaSphere Pay's jailbreak-detection routine checks only for the presence of `/Applications/Cydia.app` and whether `fork()` succeeds, both of which are trivially defeated by modern jailbreak tooling that hides Cydia-equivalent apps and patches `fork()` restrictions. TMG Security's fictional testing bypassed detection on a jailbroken test device configured with a widely available jailbreak-hiding tweak.
Business Context	Jailbreak detection is a defense-in-depth control; its weakness does not directly grant access to account data on its own; but it does mean the app cannot reliably flag a materially higher-risk execution environment to itself or its backend.
Technical Impact	The detection routine's two checks are both well-known, easily bypassed indicators; a jailbroken fictional test device with a standard jailbreak-hiding tweak installed was not flagged as jailbroken by the app.

Business Impact	The app proceeds normally on a compromised device without any elevated-risk signal, meaning defense-in-depth controls that could key off a jailbreak signal (e.g., disabling certain sensitive operations) never trigger.
Safe Proof of Concept (Summary)	*A fictional jailbroken test device with a jailbreak-hiding tweak installed was not detected as jailbroken by the app's startup check. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-020 (fictional jailbreak-detection bypass test)*
Expected Behavior	Jailbreak detection should combine multiple independent indicators (sandbox escape tests, suspicious dynamic library checks, restricted-path write tests, and where appropriate a server-side attestation signal such as DeviceCheck/App Attest) rather than relying on two easily patched checks.
Observed (Fictional) Behavior	The app relied solely on a Cydia-path check and a fork() check, both defeated by a standard jailbreak-hiding tweak.
Root Cause	The jailbreak-detection routine was implemented early in the app's history using a well-known simple pattern and was never expanded as jailbreak-hiding tooling matured.
Recommendation	Expand jailbreak detection to combine multiple independent indicators; adopt Apple's DeviceCheck/App Attest as a server-verifiable integrity signal (see the related recommendation in the Informational findings) rather than relying solely on client-side heuristics.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will expand jailbreak-detection coverage in a near-term release.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 61-90 Days | 15 December 2026 | **Remediation Planned**
Retest Recommendation	*Retest by repeating the fictional bypass test against the remediated detection routine using the same and additional jailbreak-hiding tweaks.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-RESILIENCE-1 — "The app validates the integrity of the platform."	MASWE-0051 — Root/Jailbreak Detection Not Implemented	MASTG-TEST-0240 — Jailbreak Detection in Code

IOS-019 — Sensitive Push Notification Data Exposure	**MEDIUM**
CVSS-Style Score: 4.9 (Medium) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-200 — Exposure of Sensitive Information to an Unauthorized Actor**
Affected Component	Transaction-alert push notifications
Affected iOS Component	*UNNotificationRequest content — full transaction amount and merchant name in `alert.body`, no `UNNotificationServiceExtension` redaction*
Description	Transaction-alert push notifications display the full fictional transaction amount and merchant name in the lock-screen notification body by default, with no notification-content redaction and no per-user setting to reduce lock-screen detail.
Business Context	Lock-screen notification content is visible to anyone with visual access to the device, whether or not it is unlocked, making it a common source of shoulder-surfing exposure for financial apps.
Technical Impact	The push payload's `alert.body` field contains the full transaction detail, and no `UNNotificationServiceExtension` is implemented to redact the content before display when the device is locked.
Business Impact	Transaction amounts and merchant names are visible to anyone glancing at the victim's lock screen, which for some users may reveal sensitive spending information.
Safe Proof of Concept (Summary)	*A fictional test transaction generated a lock-screen notification displaying the full amount and merchant name with the device locked. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-021 (fictional lock-screen notification capture)*
Expected Behavior	Sensitive notification content should be redacted by default on the lock screen (e.g., "New transaction — tap to view"), with full detail shown only after the device is unlocked and, ideally, an in-app authentication check has passed.
Observed (Fictional) Behavior	The full transaction amount and merchant name were shown on the lock screen by default with no redaction option.
Root Cause	The notification content was implemented to maximize at-a-glance usefulness without a corresponding privacy review of lock-screen exposure.
Recommendation	Redact sensitive fields from the default notification body and use a `UNNotificationServiceExtension` (or `UNNotificationContent.summaryArgument`) to reveal full detail only when appropriate; add a user-facing setting to control notification detail level.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will redact lock-screen notification content in a near-term release.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 61-90 Days | 15 December 2026 | **Remediation Planned**
Retest Recommendation	*Retest by repeating the fictional notification capture against the remediated build and confirming lock-screen content is redacted by default.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-PLATFORM-3 — "The app uses the user interface securely."	MASWE-0037 — Unnecessary Exposure of Sensitive Data via Notifications	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

IOS-020 — Weak Session Timeout Enforcement	MEDIUM
CVSS-Style Score: 4.8 (Medium) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-613 — Insufficient Session Expiration**
Affected Component	Client-side idle-timeout enforcement
Affected iOS Component	*`SceneDelegate.sceneDidBecomeActive` idle-timer check only, no independent server-side session-age enforcement observed*
Description	An idle-timeout value (45 minutes) is enforced client-side, re-prompting for local authentication after the app has been backgrounded past that interval, but the value is longer than appropriate for a financial application and is not independently configurable to a shorter value for higher-risk fictional account tiers.
Business Context	Idle-timeout duration directly affects the exposure window if a device is lost, stolen, or left unattended while the app remains in an authenticated state.
Technical Impact	The 45-minute timeout is applied uniformly regardless of account risk tier, and enforcement is entirely client-side — no corresponding server-side session-age check was observed during testing.
Business Impact	A lost or momentarily unattended device provides a longer-than-necessary window during which the app remains in an authenticated state without requiring re-authentication.
Safe Proof of Concept (Summary)	*Backgrounding the app for 40 minutes and returning to the foreground did not trigger a re-authentication prompt, consistent with the 45-minute client-side timeout. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-022 (fictional idle-timeout interval test)*
Expected Behavior	A shorter idle-timeout value (e.g., 5-10 minutes) appropriate to a financial application should be enforced, ideally independently configurable per account risk tier, with a corresponding server-side session-age check as defense-in-depth.
Observed (Fictional) Behavior	A 45-minute client-side-only timeout was enforced uniformly across all account types.
Root Cause	The timeout value was set during initial development based on general usability preference rather than a risk-based review specific to a financial application.
Recommendation	Reduce the default idle-timeout value and make it configurable per account risk tier; add a server-side session-age check independent of the client-side timer as defense-in-depth.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will reduce the idle-timeout value in a near-term release.*
Owner / Priority / Target / Status	**Backend/API Engineering Lead (Fictional Role)** | 61-90 Days | 15 December 2026 | **Remediation Planned**
Retest Recommendation	*Retest by repeating the fictional idle-timeout interval test against the remediated value and confirming a shorter re-authentication window.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-AUTH-1 — "The app uses secure authentication and authorization protocols and follows the relevant best practices."	MASWE-0024 — Sensitive Data Accessible After Session Termination (closest verified concept — see note)	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

IOS-021 — Insecure ATS Exception Configuration Permits Weak TLS on Secondary Endpoint	MEDIUM
CVSS-Style Score: 5.0 (Medium) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-326 — Inadequate Encryption Strength**
Affected Component	App Transport Security configuration — statement-export legacy domain exception
Affected iOS Component	*`NSAppTransportSecurity.NSExceptionDomains` entry for `docs.novasphere-example.com` with `NSExceptionMinimumTLSVersion: TLSv1.1`*
Description	The Info.plist ATS configuration carries an `NSExceptionDomains` entry for the legacy statement-export domain permitting a minimum TLS version of 1.1, rather than the 1.2+ floor enforced for all other first-party domains.
Business Context	This exception exists specifically to support the same legacy statement-export integration flagged for weak certificate validation elsewhere in this report (IOS-007), compounding that channel's overall weaker security posture.
Technical Impact	Traffic to `docs.novasphere-example.com` can negotiate down to TLS 1.1, a deprecated protocol version with known weaknesses, while the primary API domains correctly enforce TLS 1.2+.
Business Impact	The legacy statement-export channel carries a weaker minimum encryption floor than the rest of the application,

	widening the window for a network-level attacker on that specific channel.
Safe Proof of Concept (Summary)	*Static review of the Info.plist ATS configuration confirmed the `NSExceptionMinimumTLSVersion` value of 1.1 scoped to the statement-export domain. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-023 (fictional Info.plist ATS exception review)*
Expected Behavior	All first-party domains, including legacy integrations, should enforce a minimum TLS version of 1.2 or higher via ATS with no exception, consistent with the primary API domains.
Observed (Fictional) Behavior	The statement-export domain's ATS exception permitted a minimum TLS version of 1.1.
Root Cause	The ATS exception was added when the legacy document backend could not yet support TLS 1.2, and was never revisited after the backend was upgraded.
Recommendation	Remove the ATS exception for the statement-export domain and confirm the legacy backend now supports TLS 1.2+; if it does not, prioritize its upgrade or replace the integration.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will remove the ATS exception pending backend confirmation.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 61-90 Days | 15 December 2026 | **Remediation Planned**
Retest Recommendation	*Retest by reviewing the remediated Info.plist ATS configuration and confirming no domain permits a TLS version below 1.2.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-NETWORK-1 — "The app secures all network traffic according to the current best practices."	MASWE-0026 — Network Traffic Not Encrypted	MASTG-TEST-0342 — References to Weak ATS TLS Policy Exceptions in Info.plist

IOS-022 — Excessive Privacy Permission Requests Beyond Feature Requirements	**MEDIUM**
CVSS-Style Score: 4.6 (Medium) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-732 — Incorrect Permission Assignment for Critical Resource**
Affected Component	Info.plist usage-description keys and runtime permission requests
Affected iOS Component	*`NSContactsUsageDescription`, `NSMicrophoneUsageDescription` present and requested at first launch with no corresponding active feature*
Description	NovaSphere Pay requests Contacts and Microphone permissions at first launch, alongside its core Camera (check deposit) and Face ID permissions, but neither Contacts nor Microphone access is used by any currently shipping feature.
Business Context	Requesting permissions beyond active feature needs increases the app's privacy footprint and creates user trust friction, and expands the impact if either permission's underlying API were ever misused by a future SDK integration.
Technical Impact	The two unused usage-description keys and their associated runtime prompts remain present from a fictional discontinued voice-memo-support feature and an abandoned contacts-based referral feature.
Business Impact	Users are asked to grant permissions the app does not currently need, which is both a privacy-minimization gap and a source of avoidable user distrust.
Safe Proof of Concept (Summary)	*Static review of Info.plist and a walkthrough of first-launch permission prompts confirmed Contacts and Microphone permissions are requested with no corresponding active feature. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-024 (fictional Info.plist permission review)*
Expected Behavior	Only permissions backing an active, shipping feature should be requested, and only at the point of first use for that specific feature rather than in a first-launch batch.
Observed (Fictional) Behavior	Contacts and Microphone permissions were requested at first launch with no corresponding active feature.
Root Cause	Two now-discontinued features (voice-memo support, contacts-based referral) left their permission requests in place after the features themselves were removed.
Recommendation	Remove the unused `NSContactsUsageDescription` and `NSMicrophoneUsageDescription` keys and their associated runtime requests; adopt just-in-time permission requests tied to the specific feature that needs them going forward.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will remove the unused permission requests in a near-term release.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 61-90 Days | 15 December 2026 | **Remediation Planned**
Retest Recommendation	*Retest by reviewing the remediated Info.plist and confirming only actively used permissions are requested.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-PRIVACY-1 — "The app minimizes access to sensitive data and resources."	MASWE-0066 — Inadequate Permission Management	MASTG-TEST-0360 — Purpose String Accuracy for Reachable Protected Resource Access

IOS-023 — Analytics SDK Persistent Identifier Reuse Across Reinstall	MEDIUM
CVSS-Style Score: 4.7 (Medium) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-359 — Exposure of Private Personal Information to an Unauthorized Actor**
Affected Component	Third-party analytics SDK integration
Affected iOS Component	*Analytics SDK persists its own generated identifier to the Keychain (survives app deletion/reinstall) rather than honoring `identifierForVendor` reset*
Description	The bundled fictional analytics SDK stores its own tracking identifier in the Keychain rather than relying on `identifierForVendor`, so the identifier survives app deletion and reinstall, undermining a user's expectation that reinstalling the app resets their analytics identity.
Business Context	This behavior conflicts with the fictional platform's own stated privacy posture around user tracking and reduces the effectiveness of a user's decision to delete and reinstall the app as a privacy control.
Technical Impact	Because the identifier is Keychain-resident rather than tied to `identifierForVendor` (which iOS resets on full deletion of all vendor apps), it persists across the delete/reinstall cycle TMG Security tested.
Business Impact	Users who delete and reinstall the app under the belief that this resets their tracking identity are not actually granted that reset by this particular SDK.
Safe Proof of Concept (Summary)	*A fictional test device had the app deleted and reinstalled; the analytics SDK's Keychain-resident identifier was confirmed unchanged across the cycle. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-025 (fictional delete/reinstall identifier persistence test)*
Expected Behavior	Analytics and tracking identifiers should honor the platform's identity-reset boundaries (`identifierForVendor` resets on full app deletion); persisting an identifier in the Keychain specifically to survive that reset should not be done without clear user disclosure.
Observed (Fictional) Behavior	The analytics SDK's identifier persisted unchanged across app deletion and reinstall.
Root Cause	The analytics SDK's default configuration option to persist identity in the Keychain was left enabled during integration without a privacy review of its identity-reset implications.
Recommendation	Disable the analytics SDK's Keychain-persistence option and rely on `identifierForVendor` (or a fresh per-install identifier) so the identity resets consistently with the platform's own reset boundary; document the change for the privacy policy review.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will disable Keychain-based identifier persistence in the analytics SDK configuration.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 61-90 Days | 15 December 2026 | **Remediation Planned**
Retest Recommendation	*Retest by repeating the fictional delete/reinstall test against the remediated configuration and confirming the identifier resets.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-PRIVACY-2 — "The app prevents identification of the user."	MASWE-0068 — Incorrect Use of Identifiers for User Tracking	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

IOS-024 — Insecure Random Number Generation for a Security-Relevant Value	MEDIUM
CVSS-Style Score: 4.9 (Medium) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-330 — Use of Insufficiently Random Values**
Affected Component	Device-binding nonce generation
Affected iOS Component	*`arc4random_uniform` seeded indirectly from device boot time, used for the device-registration nonce rather than `SecRandomCopyBytes`*
Description	The device-binding nonce generated during fictional device registration is derived using `arc4random_uniform` combined with a value influenced by device boot time, rather than `SecRandomCopyBytes` (the platform CSPRNG appropriate for security-relevant values), producing a nonce with a narrower, more predictable generation window than intended.
Business Context	The device-binding nonce contributes to the device-registration check but is not, on its own, sufficient to bypass authentication — consistent with this finding's Medium rather than Critical/High severity.
Technical Impact	While `arc4random_uniform` is a reasonable general-purpose PRNG, combining it with a boot-time-influenced seed for a security-relevant nonce narrows its effective entropy relative to `SecRandomCopyBytes`, the platform's recommended CSPRNG for cryptographic and security-relevant use.
Business Impact	A narrower generation window for the device-binding nonce could, in combination with other weaknesses, make device-registration replay marginally easier to model than with a properly seeded CSPRNG value.

Safe Proof of Concept (Summary)	*Static review of the device-registration module's nonce-generation code confirmed use of `arc4random_uniform` with a boot-time-influenced seed rather than `SecRandomCopyBytes`. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-026 (fictional static review of nonce generation code)*
Expected Behavior	Security-relevant nonces should be generated using `SecRandomCopyBytes`, the platform's CSPRNG intended specifically for cryptographic and security-relevant randomness.
Observed (Fictional) Behavior	The device-binding nonce was generated using `arc4random_uniform` with a boot-time-influenced seed.
Root Cause	The nonce-generation code was written by a developer following a general-purpose randomness example rather than the platform's cryptographic-randomness guidance.
Recommendation	Replace the nonce-generation logic with `SecRandomCopyBytes`; audit other security-relevant random-value generation in the app for the same pattern.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will migrate nonce generation to SecRandomCopyBytes.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 61-90 Days | 15 December 2026 | **Remediation Planned**
Retest Recommendation	*Retest by reviewing the remediated nonce-generation code and confirming SecRandomCopyBytes is used.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-CRYPTO-1 — "The app employs current strong cryptography and uses it according to industry best practices."	MASWE-0012 — Improper Random Number Generation	MASTG-TEST-0311 — Insecure Random API Usage

IOS-025 — Verbose Error Handling Reveals Internal Implementation Details	**MEDIUM**
CVSS-Style Score: 4.5 (Medium) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-209 — Generation of Error Message Containing Sensitive Information**
Affected Component	Mobile API — beneficiary-management endpoint error handling
Affected iOS Component	*URLSession error-response parsing surfaces raw backend error body to a debug alert in non-App-Store build configurations*
Description	A malformed request to the beneficiary-management endpoint returns a raw fictional backend stack trace including an internal service name and a partial ORM query fragment; in the app's QA build configuration, this raw error body is surfaced directly in a debug alert rather than a generic error message.
Business Context	Verbose backend error detail is primarily a backend-side finding, but the client's QA build configuration compounds it by actively surfacing the raw error to the screen rather than the generic client-side error handling used elsewhere.
Technical Impact	The QA build configuration's error-handling path checks a build flag and, when true, displays the raw response body verbatim rather than routing it through the standard generic-error presentation used in the App Store build.
Business Impact	Internal backend implementation details are more easily observed by a fictional QA tester or anyone with access to a QA-configuration build, aiding reconnaissance of the backend's internal structure.
Safe Proof of Concept (Summary)	*Submitting a malformed request to the beneficiary-management endpoint on a QA-configuration build displayed the raw backend error body, including an internal service name, in a debug alert. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-027 (fictional malformed-request QA-build error capture)*
Expected Behavior	All build configurations, including QA, should route API error responses through the same generic, user-safe error-handling path used in the App Store build; verbose error detail should be available only in developer logs, never a user-facing alert.
Observed (Fictional) Behavior	The QA build configuration displayed the raw backend error body directly in a debug alert.
Root Cause	A QA-only debug alert was added to speed up backend integration testing and was not gated behind a stricter internal-only build flag.
Recommendation	Remove the QA-build debug alert path; route all error responses, in every build configuration, through the standard generic error-handling presentation; if verbose detail is needed for QA, log it to a developer-only diagnostic channel instead of a user-facing alert.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will remove the QA debug alert path in a near-term release.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 61-90 Days | 15 December 2026 | **Remediation Planned**
Retest Recommendation	*Retest by repeating the fictional malformed-request test against the remediated QA build and confirming only a generic error is shown.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-CODE-4 — "The app validates and sanitizes all untrusted inputs."	MASWE-0061 — Debug Artifacts Not Removed	MASTG-TEST-0358 — Implementation Details Exposure Through Logging APIs

IOS-026 — Insecure Object Persistence of Sensitive Local Cache	MEDIUM
CVSS-Style Score: 5.0 (Medium) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-312 — Cleartext Storage of Sensitive Information**
Affected Component	Local transaction-history cache (offline viewing)
Affected iOS Component	*`Codable`-backed JSON file `transaction_cache.json` written to the app's Application Support directory, unencrypted, no Data Protection class override*
Description	NovaSphere Pay caches recent fictional transaction history for offline viewing as a plain JSON file encoded via `Codable`, written to the Application Support directory without an explicit Data Protection class override, leaving it at the default `NSFileProtectionCompleteUntilFirstUserAuthentication` rather than a stricter class, and readable in cleartext once extracted from a fictional jailbroken or backup-extracted device.
Business Context	The offline-viewing feature caches meaningful transaction detail (merchant, amount, date) locally so users can browse recent activity without connectivity; this convenience data deserves the same storage-security treatment as other sensitive local data in this report.
Technical Impact	Once extracted from a fictional jailbroken test device (or an unencrypted backup after first unlock), the JSON cache file opens directly in a text editor and discloses cleartext transaction rows.
Business Impact	Recent transaction history — merchant names, amounts, and dates — is exposed to anyone with fictional jailbreak-level or backup-extraction access to the device.
Safe Proof of Concept (Summary)	*A fictional jailbroken test device's Application Support directory was read directly, and `transaction_cache.json` opened to reveal cleartext transaction rows. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-028 (fictional transaction-cache extraction)*
Expected Behavior	Cached transaction data should be encrypted at rest (e.g., wrapped with a Keychain-resident key) or, at minimum, stored with `NSFileProtectionComplete` so it is inaccessible while the device is locked.
Observed (Fictional) Behavior	The cache file used the default Data Protection class and no additional encryption, and was recoverable in cleartext from a fictional jailbroken device.
Root Cause	The offline-cache feature was implemented using a straightforward `Codable`-to-disk pattern without a follow-up storage-security review applied to the rest of the app's local data.
Recommendation	Apply `NSFileProtectionComplete` to the transaction-cache file at minimum, or wrap its contents in Keychain-backed AES-GCM encryption consistent with the app's other sensitive local caches; reduce the cached window to the minimum needed for the offline-viewing feature.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will apply stricter Data Protection to the transaction cache in a near-term release.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 61-90 Days | 15 December 2026 | **Remediation Planned**
Retest Recommendation	*Retest by repeating the fictional cache extraction against the remediated build and confirming the file is inaccessible while the device is locked.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-STORAGE-1 — "The app securely stores sensitive data."	MASWE-0001 — Sensitive Data Stored Unencrypted in Private Storage	MASTG-TEST-0300 — References to APIs for Storing Unencrypted Data in Private Storage

IOS-027 — Missing App Transport Security Domain-Level Granularity	MEDIUM
CVSS-Style Score: 4.4 (Medium) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-284 — Improper Access Control**
Affected Component	App Transport Security configuration — shared domain policy
Affected iOS Component	*Single `NSExceptionDomains`-free ATS base policy applied uniformly to first-party and third-party SDK endpoints alike*
Description	NovaSphere Pay's ATS configuration applies one shared baseline policy to all domains — first-party payments-critical endpoints and third-party SDK endpoints alike — rather than scoping stricter requirements (e.g., certificate pinning enforcement) specifically to `api.novasphere-example.com` and `auth.novasphere-example.com`.
Business Context	A single shared policy is simpler to maintain but does not let the app apply materially stricter trust requirements to its most sensitive endpoints than to a third-party analytics or mapping SDK's endpoint.
Technical Impact	While the shared baseline policy is itself reasonably strong (TLS 1.2+, no cleartext), it does not differentiate the payments-critical domains from lower-sensitivity third-party domains at the ATS/pinning configuration level.
Business Impact	The lack of domain-level granularity is a hardening gap rather than a confirmed exploitable weakness; it reduces the precision of the app's trust boundaries relative to a best-practice, per-domain configuration.

Safe Proof of Concept (Summary)	*Static review of the ATS configuration confirmed a single shared domain policy applied uniformly rather than differentiated per-domain requirements. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-029 (fictional ATS configuration review)*
Expected Behavior	First-party, payments-critical domains should carry stricter, explicitly scoped ATS and pinning requirements than lower-sensitivity third-party SDK domains.
Observed (Fictional) Behavior	A single shared ATS policy was applied to all domains without differentiation.
Root Cause	The ATS configuration was set up early in the app's development as a single global policy and was never revisited to add domain-level granularity as more third-party integrations were added.
Recommendation	Introduce domain-specific ATS exception/requirement entries scoping stricter TLS and pinning requirements to `api.novasphere-example.com` and `auth.novasphere-example.com` specifically.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will introduce domain-level ATS granularity in a near-term release.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 61-90 Days | 15 December 2026 | **Remediation Planned**
Retest Recommendation	*Retest by reviewing the remediated ATS configuration and confirming domain-specific policy granularity for payments-critical endpoints.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-NETWORK-1 — "The app secures all network traffic according to the current best practices."	MASWE-0026 — Network Traffic Not Encrypted (closest verified concept — see note; primary concern is ATS exception domain-level granularity, not a specific validation defect)	MASTG-TEST-0342 — References to Weak ATS TLS Policy Exceptions in Info.plist

73 LOW FINDINGS

The following 8 fictional Low-severity findings are hardening opportunities recommended for the 90+ day window of the illustrative roadmap.

IOS-028 — Unused and Unnecessary Privacy-Sensitive Permission Declarations	LOW
CVSS-Style Score: 3.1 (Low) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-732 — Incorrect Permission Assignment for Critical Resource**
Affected Component	Info.plist usage-description keys
Affected iOS Component	*`NSCameraUsageDescription` and `NSCalendarsUsageDescription` keys declared but corresponding capabilities not reachable from any current UI flow*
Description	Info.plist declares Camera and Calendars usage-description strings from a prior release cycle, but static and dynamic review found no current UI path that requests either permission, indicating the declarations are stale.
Business Context	Unused permission declarations widen the app's requested privacy footprint shown to users at install/App Store review time without a corresponding feature benefit.
Technical Impact	No direct technical exploitability; the risk is an inflated permission surface rather than an active vulnerability.
Business Impact	Minor trust/perception impact — privacy-conscious users or reviewers may question unused sensitive-permission requests.
Safe Proof of Concept (Summary)	*Static review of Info.plist cross-referenced against a full UI walkthrough found no reachable flow invoking camera or calendar APIs. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-030 (fictional Info.plist and UI-flow cross-reference)*
Expected Behavior	Only usage-description keys for permissions actually requested by current app functionality should be present.
Observed (Fictional) Behavior	Stale Camera and Calendars usage-description keys remain from removed or never-shipped functionality.
Root Cause	Deprecated feature flags were removed from the UI layer without a corresponding Info.plist cleanup pass.
Recommendation	Remove unused usage-description keys and confirm the corresponding entitlements/capabilities are not otherwise declared.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will clean up unused Info.plist declarations in a housekeeping release.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 90+ Days | 31 January 2027 | Remediation Planned
Retest Recommendation	*Retest by confirming the unused usage-description keys have been removed from Info.plist.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-PRIVACY-1 — "The app minimizes access to sensitive data and resources."	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

IOS-029 — Use of Deprecated iOS APIs With Superseded Secure Alternatives	LOW
CVSS-Style Score: 3.4 (Low) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-477 — Use of Obsolete Function**
Affected Component	Legacy networking and UI helper modules
Affected iOS Component	*Isolated use of `NSURLConnection`-era helper wrappers retained for backward compatibility alongside the primary `URLSession`-based networking stack*
Description	Static review of the Mach-O binary's symbol table found isolated references to deprecated networking helper APIs retained in a legacy compatibility module, alongside the app's primary modern `URLSession`-based stack.
Business Context	Deprecated APIs are typically maintained with less security scrutiny than current frameworks and may eventually be removed by Apple.
Technical Impact	No exploitable behavior was demonstrated; the deprecated paths are not reachable from the current primary networking flows exercised during testing.
Business Impact	Low direct impact; represents future maintenance and platform-compatibility risk rather than a present exploitable weakness.

Safe Proof of Concept (Summary)	*Symbol-table review of the Mach-O binary identified deprecated API references confined to a legacy compatibility module. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-031 (fictional symbol table review)*
Expected Behavior	Networking and UI code should rely exclusively on current, actively maintained iOS APIs.
Observed (Fictional) Behavior	A legacy compatibility module retains deprecated API references not reachable from primary app flows.
Root Cause	The legacy module was retained during a prior framework migration as a fallback and was never fully removed.
Recommendation	Remove the legacy compatibility module and confirm no remaining call sites depend on deprecated APIs.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will remove the legacy module in a future maintenance release.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 90+ Days | 31 January 2027 | **Remediation Planned**
Retest Recommendation	*Retest by confirming the legacy compatibility module has been removed from the binary.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-CODE-1 — "The app is built with a secure and consistent code signing configuration and its release requirements are well-documented."	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

IOS-030 — Logging Hygiene Weaknesses in Non-Sensitive Diagnostic Modules	**LOW**
CVSS-Style Score: 3.2 (Low) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-532 — Insertion of Sensitive Information into Log File**
Affected Component	Diagnostic/telemetry logging module
Affected iOS Component	*`os_log` calls in non-authentication diagnostic paths using default privacy level rather than explicit `{public}`/`{private}` annotations*
Description	Outside the higher-severity logging issue tracked as IOS-008, several lower-sensitivity diagnostic modules (UI navigation timing, cache-hit telemetry) use default-privacy `os_log` calls without explicit privacy annotations, which is inconsistent logging hygiene even though the specific values observed were low-sensitivity.
Business Context	Consistent log-privacy annotation practice reduces the chance that a future change accidentally logs sensitive data through an already-permissive code path.
Technical Impact	Values observed in these specific log statements during testing were low-sensitivity (timing/counters); the finding is about inconsistent practice rather than a confirmed leak of sensitive data.
Business Impact	Low direct impact; represents a hygiene and defense-in-depth gap.
Safe Proof of Concept (Summary)	*Fictional device console review identified diagnostic log statements without explicit privacy annotations. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-032 (fictional console log review)*
Expected Behavior	All `os_log` call sites should use explicit `{public}`/`{private}` annotations regardless of the current sensitivity of the specific value logged.
Observed (Fictional) Behavior	Several diagnostic call sites relied on default privacy behavior rather than explicit annotations.
Root Cause	Logging privacy annotation was applied ad hoc rather than enforced via lint rule or code-review checklist.
Recommendation	Adopt a lint rule or code-review checklist item requiring explicit privacy annotations on all `os_log` calls.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will introduce a logging lint rule in an upcoming sprint.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 90+ Days | 31 January 2027 | **Remediation Planned**
Retest Recommendation	*Retest by confirming a logging lint rule or checklist enforces explicit privacy annotations.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-STORAGE-2 — "The app prevents leakage of sensitive data."	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	MASTG-TEST-0297 — Sensitive Data Exposure Through Logging APIs

IOS-031 — Third-Party Dependency Governance Gaps	**LOW**

CVSS-Style Score: 3.5 (Low) — Illustrative CVSS v3.1-inspired score	CWE: CWE-1104 — Use of Unmaintained Third-Party Components
Affected Component	Swift Package Manager / CocoaPods dependency manifest
Affected iOS Component	*Third-party SDK dependency list (see Appendix M) lacking a documented review cadence or pinned-version policy for two lower-usage SDKs*
Description	Review of the dependency manifest found two lower-usage third-party SDKs (a fictional analytics helper and a fictional image-caching library) without a documented review cadence, version-pinning policy, or designated internal owner, unlike the app's core SDKs.
Business Context	Unmanaged dependencies can silently accumulate known vulnerabilities or unexpected behavior changes across upgrades.
Technical Impact	No specific vulnerable version was identified in the SDKs reviewed at time of testing; the finding concerns governance process rather than a confirmed vulnerable component.
Business Impact	Low direct impact today; elevates risk over time as unmanaged dependencies age.
Safe Proof of Concept (Summary)	*Manifest review cross-referenced against internal SDK inventory documentation found no assigned owner or review cadence for two SDKs. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-033 (fictional dependency manifest and SDK inventory review)*
Expected Behavior	All third-party SDKs should have a documented owner, review cadence, and version-pinning policy.
Observed (Fictional) Behavior	Two lower-usage SDKs lacked documented ownership and review cadence.
Root Cause	Dependency governance documentation was established for core SDKs but not extended to smaller, later-added dependencies.
Recommendation	Extend the existing dependency governance process (owner assignment, review cadence, pinned versions) to cover all third-party SDKs, including lower-usage ones.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will extend governance documentation to all SDKs.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 90+ Days | 31 January 2027 | **Remediation Planned**
Retest Recommendation	*Retest by confirming governance documentation covers all third-party SDKs in use.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-CODE-2 — "The app has effective mechanisms to detect vulnerable and outdated software components."	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

IOS-032 — Attack-Surface Hardening Opportunity — Excessive Registered URL Scheme and Extension Count	LOW
CVSS-Style Score: 3.0 (Low) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-1120 — Excessive Code Complexity**
Affected Component	Info.plist `CFBundleURLTypes` and App Extension targets
Affected iOS Component	*Six registered custom URL scheme entries and four App Extension targets, several with narrow or legacy-only usage*
Description	The app registers six custom URL scheme entries and ships four App Extension targets, several of which correspond to narrow or legacy-only functionality, broadening the overall externally reachable attack surface beyond what current functionality strictly requires.
Business Context	Each registered scheme and extension is a distinct externally reachable entry point that must be independently secured and maintained.
Technical Impact	No individually exploitable issue beyond those already tracked (IOS-005, IOS-017); this finding is about aggregate surface-area reduction as a hardening measure.
Business Impact	Low direct impact; reduces the app's overall IPC attack surface and maintenance burden if addressed.
Safe Proof of Concept (Summary)	*Static review of Info.plist and the IPA's extension bundles enumerated six URL schemes and four extensions, several with narrow current usage. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-034 (fictional Info.plist and extension bundle enumeration)*
Expected Behavior	Registered URL schemes and extensions should be limited to those required by current functionality.
Observed (Fictional) Behavior	Several registered schemes/extensions correspond to narrow or legacy-only functionality.
Root Cause	Schemes and extensions accumulated across releases without a periodic surface-area review.
Recommendation	Conduct a periodic review to retire unused or narrowly-used URL schemes and App Extensions, consolidating

	functionality where feasible.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will conduct a surface-area review in a future planning cycle.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 90+ Days | 31 January 2027 | Remediation Planned
Retest Recommendation	*Retest by confirming unused URL schemes and extensions have been retired or consolidated.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-PLATFORM-1 — "The app uses IPC mechanisms securely."	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

IOS-033 — Missing Automated Security Regression Test Coverage	**LOW**
CVSS-Style Score: 2.9 (Low) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-1120 — Excessive Code Complexity (process control gap)**
Affected Component	CI/CD pipeline (fictional)
Affected iOS Component	*No automated regression tests specifically covering prior security findings (e.g., Keychain accessibility attributes, ATS configuration)*
Description	Review of the fictional CI/CD pipeline configuration found no automated regression tests specifically targeting previously identified security-relevant configuration (e.g., Keychain accessibility attributes, ATS policy), creating risk of silent regression in future releases.
Business Context	Without automated regression coverage, previously remediated issues can silently reappear during refactors or dependency upgrades.
Technical Impact	No current exploitable weakness; the risk is regression of already-identified issues going undetected between assessments.
Business Impact	Low direct impact today; increases the likelihood that future releases reintroduce previously fixed issues.
Safe Proof of Concept (Summary)	*Review of the fictional CI/CD pipeline and test suite found no security-focused regression tests tied to prior findings. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-035 (fictional CI/CD pipeline configuration review)*
Expected Behavior	Previously identified security-relevant configuration should have corresponding automated regression tests in CI.
Observed (Fictional) Behavior	No such regression tests were present at time of testing.
Root Cause	Security testing has been performed primarily via periodic manual assessment rather than integrated into automated CI coverage.
Recommendation	Introduce automated regression tests (e.g., static checks on Keychain accessibility attributes and ATS configuration) into the CI pipeline, tied to prior finding IDs.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will scope security regression tests in a future sprint.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 90+ Days | 31 January 2027 | Remediation Planned
Retest Recommendation	*Retest by confirming automated regression tests exist for previously identified security-relevant configuration.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-CODE-4 — "The app validates and sanitizes all untrusted inputs."	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

IOS-034 — Incomplete Secure-Coding Documentation for iOS Engineering	**LOW**
CVSS-Style Score: 2.7 (Low) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-1053 — Missing Documentation for Design**
Affected Component	Internal engineering documentation (fictional)
Affected iOS Component	*No internal secure-coding guideline specific to iOS covering Keychain usage, ATS, and URL scheme/Universal Link handling*
Description	Interviews with the fictional NovaSphere mobile engineering team indicated no internal secure-coding guideline specific to iOS exists covering Keychain usage patterns, ATS configuration, or URL scheme/Universal Link handling, relying instead

	on individual engineer judgment.
Business Context	Documented secure-coding guidelines help new engineers and reviewers apply consistent security practices without relying solely on institutional knowledge.
Technical Impact	No direct technical exploitability; the finding concerns documentation and process maturity.
Business Impact	Low direct impact; increases the risk of inconsistent security practices as the engineering team grows or changes.
Safe Proof of Concept (Summary)	*Fictional interview and documentation review found no iOS-specific secure-coding guideline in the internal engineering wiki. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-036 (fictional documentation review and interview notes)*
Expected Behavior	An internal secure-coding guideline specific to iOS should exist and be referenced during code review.
Observed (Fictional) Behavior	No such guideline was found at time of testing.
Root Cause	Secure-coding practices have been transmitted informally through senior engineers rather than documented.
Recommendation	Author and publish an internal iOS secure-coding guideline covering Keychain, ATS, and IPC handling, and reference it in code-review checklists.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will draft secure-coding documentation in a future quarter.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 90+ Days | 31 January 2027 | **Remediation Planned**
Retest Recommendation	*Retest by confirming an iOS-specific secure-coding guideline has been published and referenced in review process.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-CODE-1 — "The app is built with a secure and consistent code signing configuration and its release requirements are well-documented."	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

IOS-035 — Missing CI Verification of Stack-Canary and PIE Binary Protections	**LOW**
CVSS-Style Score: 3.3 (Low) — Illustrative CVSS v3.1-inspired score	**CWE: CWE-693 — Protection Mechanism Failure**
Affected Component	CI/CD build pipeline (fictional)
Affected iOS Component	*Mach-O binary protections (stack canary, PIE) confirmed present via manual `otool` review but not automatically verified on every release build*
Description	Manual `otool -lv`/`otool -hv` review confirmed the release Mach-O binary carries standard stack-canary and PIE protections, but the fictional CI/CD pipeline does not automatically verify these flags on every build, so a future build configuration change could silently disable them without detection.
Business Context	Binary hardening flags are a baseline platform protection; silent regression would not be immediately visible without dedicated verification.
Technical Impact	No current weakness — protections were confirmed present in the assessed build; the finding is a process gap in verifying this remains true on every release.
Business Impact	Low direct impact today; risk is future silent regression going undetected.
Safe Proof of Concept (Summary)	*Manual `otool` review confirmed stack-canary and PIE flags present in the assessed build; no equivalent automated CI check was found. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-037 (fictional otool binary protection review)*
Expected Behavior	CI should automatically verify stack-canary and PIE flags on every release build and fail the build if either is missing.
Observed (Fictional) Behavior	No automated CI verification step was found; protections were confirmed only via manual review during this assessment.
Root Cause	Binary protection flags are a Swift/Xcode toolchain default and have not required active maintenance, so no dedicated CI check was ever added.
Recommendation	Add an automated CI step running `otool`-equivalent checks against each release build artifact, failing the build if stack-canary or PIE protections are absent.
Management Response	*NovaSphere engineering (fictional) has accepted this finding and will add an automated binary-protection CI check in a future sprint.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | 90+ Days | 31 January 2027 | **Remediation Planned**
Retest Recommendation	*Retest by confirming an automated CI check verifies stack-canary and PIE protections on every release build.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-RESILIENCE-4 — "The app implements device binding." (adjacent binary-hardening concept — see note)	MASWE-0045 — Compiler-Provided Security Features Not Used	MASTG-TEST-0228 / MASTG-TEST-0229 — Position Independent Code (PIC) and Stack Canaries Not Enabled

74 INFORMATIONAL FINDINGS

The following 12 fictional Informational observations carry no direct exploitation path — several document positive security controls already in place — and are recommended for the ongoing iOS-security governance and defense-in-depth backlog.

IOS-036 — Positive Observation — Multi-Factor Authentication Enforced on Account Recovery	INFORMATIONAL
CVSS-Style Score: Informational — no CVSS score applicable	**CWE: N/A — positive control observation**
Affected Component	Account recovery flow
Affected iOS Component	*Recovery flow requires email OTP plus SMS OTP before allowing credential reset*
Description	Testing confirmed the account-recovery flow consistently requires two independent out-of-band factors (email OTP and SMS OTP) before permitting a credential reset, which is a sound control against account-takeover via recovery-flow abuse.
Business Context	Recovery flows are a common account-takeover target; requiring two independent factors here is a meaningful positive control.
Technical Impact	None — positive observation.
Business Impact	None — positive observation.
Safe Proof of Concept (Summary)	*Fictional recovery-flow walkthrough confirmed both OTP factors were required in all attempted paths. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-038 (fictional recovery flow walkthrough)*
Expected Behavior	Recovery should require multiple independent factors.
Observed (Fictional) Behavior	Multiple independent factors were consistently required.
Root Cause	N/A — positive control.
Recommendation	No action required; continue to test this control in future release regression cycles.
Management Response	*Acknowledged; NovaSphere engineering (fictional) will maintain this control.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | N/A — Informational | N/A | **Observation Only**
Retest Recommendation	*No retest required; re-verify opportunistically in future assessments.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-AUTH-1 — "The app uses secure authentication and authorization protocols and follows the relevant best practices."	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

IOS-037 — Positive Observation — Strong TLS Configuration on Primary API Endpoints	INFORMATIONAL
CVSS-Style Score: Informational — no CVSS score applicable	**CWE: N/A — positive control observation**
Affected Component	Primary API network layer
Affected iOS Component	*`api.novasphere-example.com` and `auth.novasphere-example.com` negotiate TLS 1.2/1.3 only with strong cipher suites*
Description	Fictional proxy-based testing confirmed the primary API endpoints negotiate only TLS 1.2/1.3 with modern cipher suites and reject legacy protocol downgrade attempts, aside from the isolated legacy-delegate issue tracked separately as IOS-007.
Business Context	Strong baseline TLS configuration on primary endpoints reduces the overall transport-layer attack surface.
Technical Impact	None — positive observation.
Business Impact	None — positive observation.
Safe Proof of Concept (Summary)	*Fictional proxy capture confirmed TLS 1.2/1.3-only negotiation and rejection of downgrade attempts on primary endpoints. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-039 (fictional TLS negotiation capture)*
Expected Behavior	Primary endpoints should enforce modern TLS versions and cipher suites.

Observed (Fictional) Behavior	Modern TLS versions and cipher suites were enforced.
Root Cause	N/A — positive control.
Recommendation	No action required; continue to monitor as TLS best practices evolve.
Management Response	*Acknowledged; NovaSphere engineering (fictional) will maintain this configuration.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | N/A — Informational | N/A | **Observation Only**
Retest Recommendation	*No retest required; re-verify opportunistically in future assessments.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-NETWORK-1 — "The app secures all network traffic according to the current best practices."	MASWE-0028 — Insecure Identity Pinning	MASTG-TEST-0385 — Missing Certificate Pinning in ATS

IOS-038 — Recommendation — Adopt DeviceCheck or App Attest for Server-Side Device Assurance	**INFORMATIONAL**
CVSS-Style Score: Informational — no CVSS score applicable	**CWE: N/A — forward-looking recommendation**
Affected Component	Device/session binding layer
Affected iOS Component	*No use of Apple `DeviceCheck` or `App Attest` APIs identified for server-side device assurance*
Description	The app currently relies on a client-generated device-binding nonce (see IOS-024) rather than Apple's `DeviceCheck` or `App Attest` frameworks, which provide stronger, platform-attested device/session assurance signals to the backend.
Business Context	Adopting a platform-attested device-assurance mechanism would materially strengthen anti-abuse and fraud-prevention signals available to the backend.
Technical Impact	None currently — this is a forward-looking hardening recommendation, not an identified weakness beyond IOS-024.
Business Impact	None currently.
Safe Proof of Concept (Summary)	*Static and dynamic review found no `DeviceCheck`/`App Attest` integration. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-040 (fictional framework/entitlement review)*
Expected Behavior	N/A — recommendation, not a defect.
Observed (Fictional) Behavior	No DeviceCheck/App Attest integration present.
Root Cause	N/A — recommendation.
Recommendation	Evaluate integrating `DeviceCheck` or `App Attest` to supplement or replace the current client-generated nonce approach.
Management Response	*NovaSphere engineering (fictional) will evaluate DeviceCheck/App Attest adoption in a future roadmap review.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | N/A — Informational | N/A | **Observation Only**
Retest Recommendation	*No retest required; track as a roadmap item.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-RESILIENCE-4 — "The app implements device binding."	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

IOS-039 — Recommendation — Evaluate Binary Obfuscation for Sensitive Business Logic	**INFORMATIONAL**
CVSS-Style Score: Informational — no CVSS score applicable	**CWE: N/A — forward-looking recommendation**
Affected Component	Mach-O binary
Affected iOS Component	*No symbol-name obfuscation or control-flow obfuscation applied to sensitive payments-logic modules*
Description	Static review of the Mach-O binary's symbol table found readable Swift type and method names throughout, including in payments-logic modules, with no obfuscation applied.

Business Context	Obfuscation raises the effort required for static reverse engineering of sensitive business logic, complementing (not replacing) runtime protections.
Technical Impact	None currently identified beyond the general reverse-engineering exposure inherent to any unobfuscated binary.
Business Impact	None currently.
Safe Proof of Concept (Summary)	*Symbol-table review found unobfuscated Swift type/method names in sensitive modules. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-041 (fictional symbol table review)*
Expected Behavior	N/A — recommendation, not a defect.
Observed (Fictional) Behavior	No obfuscation was applied to sensitive modules.
Root Cause	N/A — recommendation.
Recommendation	Evaluate applying symbol and control-flow obfuscation to payments-critical business logic modules.
Management Response	*NovaSphere engineering (fictional) will evaluate obfuscation tooling in a future roadmap review.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | N/A — Informational | N/A | **Observation Only**
Retest Recommendation	*No retest required; track as a roadmap item.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-RESILIENCE-2 — "The app implements a mechanism to detect and respond to tampering of executable files and critical data."	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

IOS-040 — Recommendation — Formalize Secrets Management for Build-Time Configuration	**INFORMATIONAL**
CVSS-Style Score: Informational — no CVSS score applicable	**CWE: N/A — forward-looking recommendation**
Affected Component	Build configuration / CI pipeline (fictional)
Affected iOS Component	*Build-time configuration values injected via xcconfig files without a centralized secrets-management step*
Description	Beyond the confirmed hardcoded-key finding (IOS-009), review of the broader build configuration process found no centralized secrets-management tooling (e.g., a dedicated secrets vault integrated into CI) governing how build-time configuration values are sourced and injected.
Business Context	A centralized secrets-management process reduces the likelihood of future hardcoded-credential issues recurring.
Technical Impact	None beyond IOS-009, which is already tracked separately.
Business Impact	None beyond IOS-009.
Safe Proof of Concept (Summary)	*Review of the fictional build configuration process found no centralized secrets-management tooling in use. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-042 (fictional build configuration review)*
Expected Behavior	N/A — recommendation, not a defect.
Observed (Fictional) Behavior	No centralized secrets-management tooling was in use.
Root Cause	N/A — recommendation.
Recommendation	Adopt a centralized secrets-management solution integrated into the CI/CD pipeline for all build-time configuration values.
Management Response	*NovaSphere engineering (fictional) will evaluate secrets-management tooling alongside the IOS-009 remediation.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | N/A — Informational | N/A | **Observation Only**
Retest Recommendation	*No retest required; track alongside IOS-009 remediation.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-STORAGE-1 — "The app securely stores sensitive data."	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

IOS-041 — Recommendation — Add Step-Up Biometric Re-Authentication for High-Value Transactions	INFORMATIONAL
CVSS-Style Score: Informational — no CVSS score applicable	**CWE: N/A — forward-looking recommendation**
Affected Component	Transaction authorization flow
Affected iOS Component	*Biometric prompt occurs once at app unlock; high-value transfers do not trigger a second, transaction-scoped biometric prompt*
Description	The app currently gates biometric authentication only at app unlock; a session-active user can initiate high-value transfers without a second, transaction-scoped biometric confirmation, which is a hardening opportunity beyond current baseline behavior.
Business Context	Step-up authentication on high-value actions is a common defense-in-depth control in payments applications.
Technical Impact	None currently identified as an active exploit path; this is a defense-in-depth recommendation.
Business Impact	None currently.
Safe Proof of Concept (Summary)	*Fictional transaction-flow walkthrough confirmed no additional biometric prompt for high-value transfers within an active session. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-043 (fictional transaction flow walkthrough)*
Expected Behavior	N/A — recommendation, not a defect.
Observed (Fictional) Behavior	No transaction-scoped step-up biometric prompt was present.
Root Cause	N/A — recommendation.
Recommendation	Add a transaction-scoped biometric (or passcode) re-authentication prompt for transfers above a defined value threshold.
Management Response	*NovaSphere engineering (fictional) will evaluate step-up authentication for high-value transfers in a future roadmap review.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | N/A — Informational | N/A | **Observation Only**
Retest Recommendation	*No retest required; track as a roadmap item.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-AUTH-2 — "The app performs local authentication securely according to the platform best practices."	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

IOS-042 — Recommendation — Formalize Third-Party SDK Inventory Tracking	INFORMATIONAL
CVSS-Style Score: Informational — no CVSS score applicable	**CWE: N/A — forward-looking recommendation**
Affected Component	SDK inventory/documentation process (fictional)
Affected iOS Component	*SDK inventory (Appendix M) was compiled for this assessment rather than maintained continuously by engineering*
Description	The comprehensive third-party SDK inventory produced for this report (Appendix M) was compiled specifically for this assessment; no equivalent continuously-maintained inventory was found within the fictional engineering organization.
Business Context	A continuously maintained SDK inventory supports faster response to newly disclosed third-party vulnerabilities.
Technical Impact	None currently.
Business Impact	None currently.
Safe Proof of Concept (Summary)	*Interview confirmed the SDK inventory was assembled for this assessment rather than maintained on an ongoing basis. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-044 (fictional interview notes)*
Expected Behavior	N/A — recommendation, not a defect.
Observed (Fictional) Behavior	No continuously maintained SDK inventory was found.
Root Cause	N/A — recommendation.
Recommendation	Adopt the assessment's SDK inventory (Appendix M) as a living document maintained by engineering going forward.
Management Response	*NovaSphere engineering (fictional) will adopt Appendix M as a starting point for an ongoing SDK inventory.*

Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | N/A — Informational | N/A | **Observation Only**
Retest Recommendation	*No retest required; track as a roadmap item.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-CODE-2 — "The app has effective mechanisms to detect vulnerable and outdated software components."	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

IOS-043 — Recommendation — Review Crash-Reporting Payloads for Incidental PII	**INFORMATIONAL**
CVSS-Style Score: Informational — no CVSS score applicable	**CWE: N/A — forward-looking recommendation**
Affected Component	Crash-reporting SDK integration
Affected iOS Component	*Fictional crash-reporting SDK configured with default payload capture; no documented review of incidental PII in captured breadcrumbs*
Description	The fictional crash-reporting SDK is configured with default breadcrumb/payload capture; no documented periodic review was found confirming captured breadcrumbs are free of incidental PII (e.g., account identifiers appearing in UI-state breadcrumbs).
Business Context	Crash-reporting pipelines are a common inadvertent channel for PII leakage into third-party vendor systems.
Technical Impact	No specific PII leakage was confirmed during this assessment; this is a recommendation for a periodic review process.
Business Impact	None currently confirmed.
Safe Proof of Concept (Summary)	*Configuration review found default breadcrumb capture with no documented periodic PII review process. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-045 (fictional crash-reporting SDK configuration review)*
Expected Behavior	N/A — recommendation, not a defect.
Observed (Fictional) Behavior	No periodic PII review process was documented for crash-reporting payloads.
Root Cause	N/A — recommendation.
Recommendation	Establish a periodic review of crash-reporting breadcrumb/payload content for incidental PII, with redaction rules as needed.
Management Response	*NovaSphere engineering (fictional) will establish a periodic crash-reporting content review process.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | N/A — Informational | N/A | **Observation Only**
Retest Recommendation	*No retest required; track as a roadmap item.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-PRIVACY-2 — "The app informs the user prior to any collection of personal identifiable information (PII) and asks for permission."	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

IOS-044 — Recommendation — Establish a Secure-SDLC Checkpoint for Mobile Releases	**INFORMATIONAL**
CVSS-Style Score: Informational — no CVSS score applicable	**CWE: N/A — forward-looking recommendation**
Affected Component	Release process (fictional)
Affected iOS Component	*No documented security sign-off checkpoint identified within the mobile release process*
Description	Interviews with the fictional release management function found no formal, documented security sign-off checkpoint gating mobile releases, relying instead on periodic external assessments such as this one.
Business Context	A recurring internal security checkpoint complements periodic external assessments by catching issues closer to the point of introduction.
Technical Impact	None currently.
Business Impact	None currently.

Safe Proof of Concept (Summary)	*Fictional interview with release management found no documented security sign-off checkpoint in the release process. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-046 (fictional interview notes)*
Expected Behavior	N/A — recommendation, not a defect.
Observed (Fictional) Behavior	No documented security sign-off checkpoint was found.
Root Cause	N/A — recommendation.
Recommendation	Establish a lightweight, documented security checkpoint (e.g., a release-gate checklist) as part of the standard mobile release process.
Management Response	*NovaSphere engineering (fictional) will pilot a release-gate security checklist in an upcoming cycle.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | N/A — Informational | N/A | **Observation Only**
Retest Recommendation	*No retest required; track as a roadmap item.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-CODE-1 — "The app is built with a secure and consistent code signing configuration and its release requirements are well-documented."	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

IOS-045 — Observation — Siri Shortcuts and App Intents Surface Is Appropriately Restricted	**INFORMATIONAL**
CVSS-Style Score: Informational — no CVSS score applicable	**CWE: N/A — positive control observation**
Affected Component	SiriKit / App Intents integration
Affected iOS Component	*Exposed App Intents are limited to balance-check and nearby-ATM lookup; no funds-movement intent is exposed to Siri/Shortcuts*
Description	Review of the app's App Intents/SiriKit integration confirmed only low-sensitivity read-only intents (balance check, nearby-ATM lookup) are exposed to Siri and Shortcuts, with no funds-movement or account-modification intent reachable through this surface, including via AI agent-style invocation of exposed App Intents.
Business Context	Limiting voice/AI-agent-reachable intents to read-only, low-sensitivity actions is a sound design choice for a payments app.
Technical Impact	None — positive observation.
Business Impact	None — positive observation.
Safe Proof of Concept (Summary)	*Fictional App Intents enumeration confirmed only read-only intents were exposed. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-047 (fictional App Intents enumeration)*
Expected Behavior	Sensitive account-modification actions should not be exposed via Siri/Shortcuts/App Intents.
Observed (Fictional) Behavior	Only low-sensitivity read-only intents were exposed.
Root Cause	N/A — positive control.
Recommendation	No action required; continue to review new App Intents against this principle before exposing them.
Management Response	*Acknowledged; NovaSphere engineering (fictional) will maintain this restriction as new intents are added.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | N/A — Informational | N/A | **Observation Only**
Retest Recommendation	*No retest required; re-verify opportunistically as new App Intents are added.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-PLATFORM-1 — "The app uses IPC mechanisms securely."	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

IOS-046 — Recommendation — Increase Dependency-Scanning Cadence for Third-Party SDKs	**INFORMATIONAL**

CVSS-Style Score: Informational — no CVSS score applicable	CWE: N/A — forward-looking recommendation
Affected Component	Dependency scanning process (fictional)
Affected iOS Component	*Fictional dependency vulnerability scanning runs on an ad hoc basis rather than a fixed recurring schedule*
Description	Interviews indicated the fictional dependency vulnerability scanning process runs ad hoc rather than on a fixed recurring schedule (e.g., weekly or per-build), which relates to but is distinct from the governance gap tracked as IOS-031.
Business Context	A fixed recurring scanning cadence reduces the window between a new CVE disclosure and its detection in the app's dependency tree.
Technical Impact	None currently.
Business Impact	None currently.
Safe Proof of Concept (Summary)	*Fictional interview confirmed dependency scanning occurs on an ad hoc rather than fixed schedule. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-048 (fictional interview notes)*
Expected Behavior	N/A — recommendation, not a defect.
Observed (Fictional) Behavior	Dependency scanning occurs ad hoc rather than on a fixed schedule.
Root Cause	N/A — recommendation.
Recommendation	Move dependency vulnerability scanning to a fixed recurring schedule integrated into CI (e.g., per-build or weekly).
Management Response	*NovaSphere engineering (fictional) will move to a fixed scanning cadence in an upcoming sprint.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | N/A — Informational | N/A | **Observation Only**
Retest Recommendation	*No retest required; track as a roadmap item.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-CODE-2 — "The app has effective mechanisms to detect vulnerable and outdated software components."	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

IOS-047 — Recommendation — Document Provisioning Profile and Signing Certificate Lifecycle Management	INFORMATIONAL
CVSS-Style Score: Informational — no CVSS score applicable	**CWE: N/A — forward-looking recommendation**
Affected Component	Code signing / provisioning process (fictional)
Affected iOS Component	*No documented lifecycle process found for signing certificate rotation or provisioning profile renewal and revocation*
Description	Review of the fictional code signing and provisioning process found the current distribution certificate and provisioning profile to be validly configured and appropriately scoped, but no documented lifecycle process for rotation, renewal, or emergency revocation was identified.
Business Context	A documented signing lifecycle process reduces the risk of signing-related release disruptions and clarifies revocation procedures in case of key compromise.
Technical Impact	None currently — the current configuration was found valid and appropriately scoped.
Business Impact	None currently.
Safe Proof of Concept (Summary)	*Fictional codesign/provisioning profile review confirmed valid current configuration but no documented lifecycle process. [ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE]*
Evidence Reference	*EV-IOS-049 (fictional codesign and provisioning profile review)*
Expected Behavior	N/A — recommendation, not a defect.
Observed (Fictional) Behavior	No documented signing/provisioning lifecycle process was found.
Root Cause	N/A — recommendation.
Recommendation	Document the signing certificate and provisioning profile lifecycle, including rotation, renewal, and emergency revocation procedures.
Management Response	*NovaSphere engineering (fictional) will document the signing lifecycle process in an upcoming release cycle.*
Owner / Priority / Target / Status	**Mobile Engineering Lead (Fictional Role)** | N/A — Informational | N/A | **Observation Only**
Retest Recommendation	*No retest required; track as a roadmap item.*

MASVS MAPPING	MASWE MAPPING	MASTG TEST MAPPING
MASVS-CODE-1 — "The app is built with a secure and consistent code signing configuration and its release requirements are well-documented."	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.

75 MANAGEMENT ACTION PLAN

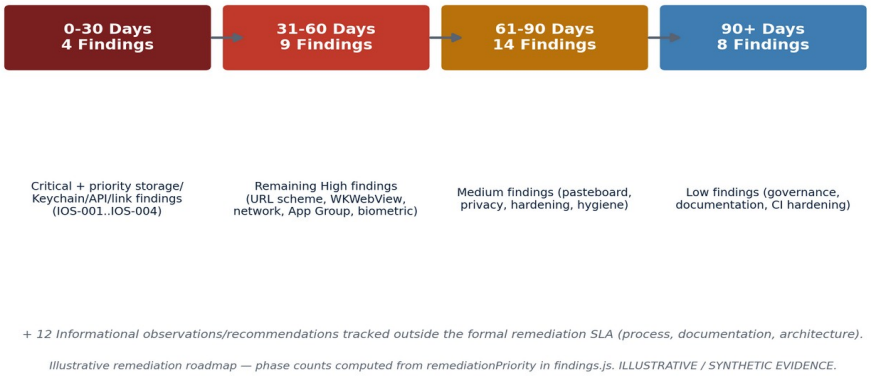
The table below tracks the fictional management response, ownership, and validation approach for all 47 findings, ensuring every finding identified in Sections 70-74 appears here exactly once.

Finding ID	Owner	Priority	Target Date	Status	Validation Method	Remediation (Summary)
IOS-001	Mobile Engineering Lead	Immediate — 0-30 Days	10 October 2026	Remediation In Progress	Full or Targeted Retest	Migrate all session/token persistence to the iOS Keychain w…
IOS-002	Mobile Engineering Lead	Immediate — 0-30 Days	10 October 2026	Remediation In Progress	Full or Targeted Retest	Re-create the biometric-vault Keychain item using kSecAttrA…
IOS-003	Backend/API Engineering L…	Immediate — 0-30 Days	24 September 2026	Remediation In Progress	Full or Targeted Retest	Add a mandatory server-side check on every account-scoped m…
IOS-004	Mobile Engineering Lead	Immediate — 0-30 Days	10 October 2026	Remediation In Progress	Full or Targeted Retest	Remove the silent auto-confirm path from the Universal Link…
IOS-005	Mobile Engineering Lead	31-60 Days	15 November 2026	Remediation Planned	Full or Targeted Retest	Migrate the OAuth callback to a Universal Link under the ve…
IOS-006	Mobile Engineering Lead	31-60 Days	15 November 2026	Remediation Planned	Full or Targeted Retest	Remove the session-token and device-identifier getters from…
IOS-007	Mobile Engineering Lead	31-60 Days	15 November 2026	Remediation Planned	Full or Targeted Retest	Fix the custom URLSessionDelegate to enforce SecTrustEvalua…
IOS-008	Mobile Engineering Lead	31-60 Days	15 November 2026	Remediation Planned	Full or Targeted Retest	Remove all sensitive-value logging from the networking and…
IOS-009	Mobile Engineering Lead	31-60 Days	15 November 2026	Remediation Planned	Full or Targeted Retest	Move the mapping SDK key behind a server-side proxy endpoin…
IOS-010	Mobile Engineering Lead	31-60 Days	15 November 2026	Remediation Planned	Full or Targeted Retest	Reduce the App Group payload to the minimum display-necessa…
IOS-011	Mobile Engineering Lead	31-60 Days	15 November 2026	Remediation Planned	Full or Targeted Retest	Create a separate Keychain access group scoped to the main…
IOS-013	Mobile Engineering Lead	31-60 Days	15 November 2026	Remediation Planned	Full or Targeted Retest	Rework the unlock gate to require successfully reading a Ke…
IOS-014	Backend/API Engineering L…	31-60 Days	15 November 2026	Remediation Planned	Full or Targeted Retest	Add a server-side token-revocation call to the logout flow,…
IOS-012	Mobile Engineering Lead	61-90 Days	15 December 2026	Remediation Planned	Full or Targeted Retest	Switch to `UIPasteboard.setItems(_:optio ns:)` with `.expira…
IOS-015	Mobile Engineering Lead	61-90 Days	15 December 2026	Remediation Planned	Full or Targeted Retest	Extend the existing privacy-overlay pattern to every screen…
IOS-016	Mobile Engineering Lead	61-90 Days	15 December 2026	Remediation Planned	Full or Targeted Retest	Disable `UIFileSharingEnabled` for the Documents directory…
IOS-017	Mobile Engineering Lead	61-90 Days	15 December 2026	Remediation Planned	Full or Targeted Retest	Refactor the share extension to receive only the selected t…
IOS-018	Mobile Engineering Lead	61-90 Days	15 December 2026	Remediation Planned	Full or Targeted Retest	Expand jailbreak detection to combine multiple independent…
IOS-019	Mobile Engineering Lead	61-90 Days	15 December 2026	Remediation Planned	Full or Targeted Retest	Redact sensitive fields from the default notification body…
IOS-020	Backend/API Engineering L…	61-90 Days	15 December 2026	Remediation Planned	Full or Targeted Retest	Reduce the default idle-timeout value and make it configura…
IOS-021	Mobile Engineering Lead	61-90 Days	15 December 2026	Remediation Planned	Full or Targeted Retest	Remove the ATS exception for the statement-export domain an…
IOS-022	Mobile Engineering Lead	61-90 Days	15 December 2026	Remediation Planned	Full or Targeted Retest	Remove the unused `NSContactsUsageDescription` and `NSMicro…
IOS-023	Mobile Engineering Lead	61-90 Days	15 December 2026	Remediation Planned	Full or Targeted Retest	Disable the analytics SDK's Keychain-persistence option and…
IOS-024	Mobile Engineering Lead	61-90 Days	15 December 2026	Remediation Planned	Full or Targeted Retest	Replace the nonce-generation logic with `SecRandomCopyBytes…
IOS-025	Mobile Engineering Lead	61-90 Days	15 December 2026	Remediation Planned	Full or Targeted Retest	Remove the QA-build debug alert path; route all error respo…

Finding ID	Owner	Priority	Target Date	Status	Validation Method	Remediation (Summary)
IOS-026	Mobile Engineering Lead	61-90 Days	15 December 2026	Remediation Planned	Full or Targeted Retest	Apply `NSFileProtectionComplete` to the transaction-cache f…
IOS-027	Mobile Engineering Lead	61-90 Days	15 December 2026	Remediation Planned	Full or Targeted Retest	Introduce domain-specific ATS exception/requirement entries…
IOS-028	Mobile Engineering Lead	90+ Days	31 January 2027	Remediation Planned	Full or Targeted Retest	Remove unused usage-description keys and confirm the corres…
IOS-029	Mobile Engineering Lead	90+ Days	31 January 2027	Remediation Planned	Full or Targeted Retest	Remove the legacy compatibility module and confirm no remai…
IOS-030	Mobile Engineering Lead	90+ Days	31 January 2027	Remediation Planned	Full or Targeted Retest	Adopt a lint rule or code-review checklist item requiring e…
IOS-031	Mobile Engineering Lead	90+ Days	31 January 2027	Remediation Planned	Full or Targeted Retest	Extend the existing dependency governance process (owner as…
IOS-032	Mobile Engineering Lead	90+ Days	31 January 2027	Remediation Planned	Full or Targeted Retest	Conduct a periodic review to retire unused or narrowly-used…
IOS-033	Mobile Engineering Lead	90+ Days	31 January 2027	Remediation Planned	Full or Targeted Retest	Introduce automated regression tests (e.g., static checks o…
IOS-034	Mobile Engineering Lead	90+ Days	31 January 2027	Remediation Planned	Full or Targeted Retest	Author and publish an internal iOS secure-coding guideline…
IOS-035	Mobile Engineering Lead	90+ Days	31 January 2027	Remediation Planned	Full or Targeted Retest	Add an automated CI step running `otool`-equivalent checks…
IOS-036	Mobile Engineering Lead	N/A — Informational	N/A	Observation Only	Documentation / Config Review	No action required; continue to test this control in future…
IOS-037	Mobile Engineering Lead	N/A — Informational	N/A	Observation Only	Documentation / Config Review	No action required; continue to monitor as TLS best practic…
IOS-038	Mobile Engineering Lead	N/A — Informational	N/A	Observation Only	Documentation / Config Review	Evaluate integrating `DeviceCheck` or `App Attest` to suppl…
IOS-039	Mobile Engineering Lead	N/A — Informational	N/A	Observation Only	Documentation / Config Review	Evaluate applying symbol and control-flow obfuscation to pa…
IOS-040	Mobile Engineering Lead	N/A — Informational	N/A	Observation Only	Documentation / Config Review	Adopt a centralized secrets-management solution integrated…
IOS-041	Mobile Engineering Lead	N/A — Informational	N/A	Observation Only	Documentation / Config Review	Add a transaction-scoped biometric (or passcode) re-authent…
IOS-042	Mobile Engineering Lead	N/A — Informational	N/A	Observation Only	Documentation / Config Review	Adopt the assessment's SDK inventory (Appendix M) as a livi…
IOS-043	Mobile Engineering Lead	N/A — Informational	N/A	Observation Only	Documentation / Config Review	Establish a periodic review of crash-reporting breadcrumb/p…
IOS-044	Mobile Engineering Lead	N/A — Informational	N/A	Observation Only	Documentation / Config Review	Establish a lightweight, documented security checkpoint (e.…
IOS-045	Mobile Engineering Lead	N/A — Informational	N/A	Observation Only	Documentation / Config Review	No action required; continue to review new App Intents agai…
IOS-046	Mobile Engineering Lead	N/A — Informational	N/A	Observation Only	Documentation / Config Review	Move dependency vulnerability scanning to a fixed recurring…
IOS-047	Mobile Engineering Lead	N/A — Informational	N/A	Observation Only	Documentation / Config Review	Document the signing certificate and provisioning profile l…

76 REMEDIATION ROADMAP

The illustrative sample remediation roadmap below sequences all 47 fictional findings across four phases, directly derived from each finding's remediationPriority field, prioritizing the four Critical findings and the nine High findings first.



ILLUSTRATIVE SAMPLE REMEDIATION ROADMAP — fictional sequencing and dates.

Immediate — 0-30 Days (4 findings)

All four Critical findings — concentrated in insecure local/Keychain token storage, broken mobile-API object-level authorization, and a silent Universal Link account-modification flow — require immediate remediation given their direct path to fictional account or funds compromise.

- Sensitive Authentication Token Exposure Through Insecure Local Storage — IOS-001 (CRITICAL)
- Insecure Keychain Access Control Allows Unauthorized Credential Access — IOS-002 (CRITICAL)
- Broken Authorization in Mobile API Workflow — IOS-003 (CRITICAL)
- Insecure Universal Link / Deep-Link Flow Enables Sensitive Account Action — IOS-004 (CRITICAL)

31-60 Days (9 findings)

The nine High findings, spanning custom URL scheme handling, the WKWebView JavaScript bridge, certificate/trust-evaluation weaknesses, log-based token exposure, a hardcoded SDK key, App Group and Keychain access-group scoping, biometric-gate enforcement, and refresh-token revocation, should be remediated in this window.

- Insecure Custom URL Scheme Enables Authentication or Account Flow Hijacking — IOS-005 (HIGH)
- WKWebView JavaScript Bridge Exposes Sensitive Native Functionality — IOS-006 (HIGH)
- Weak Certificate Validation / Trust Evaluation Logic — IOS-007 (HIGH)
- Sensitive Data Exposure Through Application Logs — IOS-008 (HIGH)
- Hardcoded API Secret / Credential Material in iOS Binary — IOS-009 (HIGH)
- Insecure App Group Data Sharing — IOS-010 (HIGH)
- Keychain Access Group Misconfiguration — IOS-011 (HIGH)
- Biometric Authentication Bypass / Improper LocalAuthentication Enforcement — IOS-013 (HIGH)
- Missing Refresh Token Revocation After Logout — IOS-014 (HIGH)

61-90 Days (14 findings)

The fourteen Medium findings — defense-in-depth and platform-hardening gaps across pasteboard handling, background-snapshot exposure, file-sharing scope, extension data isolation, jailbreak-detection strength, notification content, session timeout, ATS granularity, permissions, analytics identifiers, random-number generation, error handling, and local cache protection — are recommended for this window.

- Sensitive Data Exposure Through Pasteboard — IOS-012 (MEDIUM)
- Sensitive Data Exposure Through Background App Snapshot — IOS-015 (MEDIUM)
- Insecure Document Interaction / File Sharing Configuration — IOS-016 (MEDIUM)

- Improper App Extension Data Isolation — IOS-017 (MEDIUM)
- Weak Jailbreak / Runtime Integrity Protection — IOS-018 (MEDIUM)
- Sensitive Push Notification Data Exposure — IOS-019 (MEDIUM)
- Weak Session Timeout Enforcement — IOS-020 (MEDIUM)
- Insecure ATS Exception Configuration Permits Weak TLS on Secondary Endpoint — IOS-021 (MEDIUM)
- Excessive Privacy Permission Requests Beyond Feature Requirements — IOS-022 (MEDIUM)
- Analytics SDK Persistent Identifier Reuse Across Reinstall — IOS-023 (MEDIUM)
- Insecure Random Number Generation for a Security-Relevant Value — IOS-024 (MEDIUM)
- Verbose Error Handling Reveals Internal Implementation Details — IOS-025 (MEDIUM)
- Insecure Object Persistence of Sensitive Local Cache — IOS-026 (MEDIUM)
- Missing App Transport Security Domain-Level Granularity — IOS-027 (MEDIUM)

90+ Days (20 findings)

The eight Low-severity hardening items and the twelve Informational observations/recommendations — covering unused permissions, deprecated APIs, logging hygiene, dependency governance, attack-surface consolidation, CI verification gaps, and forward-looking process recommendations (DeviceCheck/App Attest adoption, obfuscation, secrets management, step-up authentication, SDK inventory, crash-reporting review, secure-SDLC checkpoints, dependency-scanning cadence, and signing-lifecycle documentation) round out the ongoing governance backlog.

- Unused and Unnecessary Privacy-Sensitive Permission Declarations — IOS-028 (LOW)
- Use of Deprecated iOS APIs With Superseded Secure Alternatives — IOS-029 (LOW)
- Logging Hygiene Weaknesses in Non-Sensitive Diagnostic Modules — IOS-030 (LOW)
- Third-Party Dependency Governance Gaps — IOS-031 (LOW)
- Attack-Surface Hardening Opportunity — Excessive Registered URL Scheme and Extension Count — IOS-032 (LOW)
- Missing Automated Security Regression Test Coverage — IOS-033 (LOW)
- Incomplete Secure-Coding Documentation for iOS Engineering — IOS-034 (LOW)
- Missing CI Verification of Stack-Canary and PIE Binary Protections — IOS-035 (LOW)
- Positive Observation — Multi-Factor Authentication Enforced on Account Recovery — IOS-036 (INFORMATIONAL)
- Positive Observation — Strong TLS Configuration on Primary API Endpoints — IOS-037 (INFORMATIONAL)
- Recommendation — Adopt DeviceCheck or App Attest for Server-Side Device Assurance — IOS-038 (INFORMATIONAL)
- Recommendation — Evaluate Binary Obfuscation for Sensitive Business Logic — IOS-039 (INFORMATIONAL)
- Recommendation — Formalize Secrets Management for Build-Time Configuration — IOS-040 (INFORMATIONAL)
- Recommendation — Add Step-Up Biometric Re-Authentication for High-Value Transactions — IOS-041 (INFORMATIONAL)
- Recommendation — Formalize Third-Party SDK Inventory Tracking — IOS-042 (INFORMATIONAL)
- Recommendation — Review Crash-Reporting Payloads for Incidental PII — IOS-043 (INFORMATIONAL)
- Recommendation — Establish a Secure-SDLC Checkpoint for Mobile Releases — IOS-044 (INFORMATIONAL)
- Observation — Siri Shortcuts and App Intents Surface Is Appropriately Restricted — IOS-045 (INFORMATIONAL)
- Recommendation — Increase Dependency-Scanning Cadence for Third-Party SDKs — IOS-046 (INFORMATIONAL)
- Recommendation — Document Provisioning Profile and Signing Certificate Lifecycle Management — IOS-047 (INFORMATIONAL)

ILLUSTRATIVE SAMPLE REMEDIATION ROADMAP — sequencing, owners, and dates are fictional.

77 RETEST RECOMMENDATIONS

TMG Security recommends a structured retest following remediation, applying the methodology below. This section describes the recommended retest approach for this fictional engagement; it does not state that remediation has actually occurred.

Recommended Retest Scope

- Full retest of all four Critical findings (IOS-001 through IOS-004) and all nine High findings (IOS-005 through IOS-014), including a fresh IPA build and repeated static/dynamic analysis pass.
- Targeted retest of each Medium finding against its specific affected screen, Keychain item, App Group container, or mobile-API endpoint, following each finding's individual retestRecommendation field.
- Confirmation-only review of Low and Informational items, batched into the next general assessment cycle where full retest is not independently warranted.

Retest Methodology

- Evidence Review — confirm that a code or configuration change addressing the finding's root cause has been merged and reflected in a newly built fictional IPA (version/build number incremented from 6.4.2 / 6402).
- Original PoC Reproduction — attempt the original safe PoC steps (static review, Keychain/filesystem inspection on a fictional jailbroken device, or mobile-API request, as applicable) to confirm the previously observed fictional behavior no longer occurs.
- Info.plist, Entitlements & URL Scheme Re-Review — re-run the Info.plist, entitlements, and URL scheme/Universal Link analysis (Appendices C-F) to confirm no adjacent configuration regressed.
- Static Re-Analysis — re-decompile the new IPA build (Mach-O binary, symbol table, strings) and re-run the relevant MASTG-aligned test cases from Appendix A for the affected finding.
- Dynamic Re-Testing — re-run the affected finding's dynamic test on a fresh fictional jailbroken and non-jailbroken device pair, consistent with the original test-account register.
- Network Regression — for network/certificate findings (IOS-007, IOS-021, IOS-027), re-validate TLS/ATS configuration and certificate trust-evaluation behavior through the interception-proxy methodology used earlier in this report.
- Resilience Regression — for resilience findings (IOS-018), independently re-verify jailbreak-detection behavior under both jailbreak-hiding and runtime-instrumentation conditions, not only a baseline jailbroken device.
- Keychain & Local Authentication Regression — for Keychain and biometric findings (IOS-001, IOS-002, IOS-011, IOS-013), confirm the remediated accessibility class, SecAccessControl policy, and Keychain access-group scoping directly, rather than relying on functional testing alone.
- MASVS Coverage Re-Check — confirm the retested finding's MASVS control now shows a Pass status in an updated Appendix H coverage matrix.
- Closure Criteria — a finding is marked Closed only after reproduction confirms remediation and regression testing raises no new concern.

This retest methodology is illustrative. This report does not claim that any finding has been remediated or retested.

78 FINAL CONCLUSION

Illustrative Security Assessment Conclusion

"NovaSphere Pay for iOS demonstrates a strong baseline in several areas — enforced multi-factor authentication on account recovery, TLS 1.2/1.3-only negotiation on the primary API endpoints, and an appropriately restricted Siri Shortcuts/App Intents surface — but the fictional assessment identified meaningful Keychain configuration, Universal Link/deep-link handling, and mobile-API authorization weaknesses, concentrated in four Critical findings, that require immediate, prioritized remediation."

The four Critical and nine High fictional findings in this sample report require immediate attention and should be the organization's first priority, given their concentration around local/Keychain token storage, Universal Link and custom URL scheme handling, WKWebView JavaScript bridge exposure, and mobile-API object-level authorization — the classic iOS-specific trust boundaries where a mobile client's Keychain, IPC surface, and App Extension model intersect with server-side enforcement. The fourteen Medium findings should be integrated into the standard remediation backlog and addressed within the illustrative 61-90 day window. The eight Low and twelve Informational findings support defense-in-depth, resilience, and mobile-security governance improvements and do not represent an immediate exploitation path in this fictional sample, but should not be indefinitely deferred.

TMG Security's simulated recommendation is that NovaSphere Digital Technologies invest specifically in (1) migrating all locally stored session and refresh-token material to Keychain-backed storage with an appropriate accessibility class and biometry-bound SecAccessControl, (2) requiring explicit in-app confirmation for every sensitive account action reachable via a Universal Link or custom URL scheme, and (3) closing the object-level authorization gap in the mobile API's funds-transfer confirmation workflow — the three highest-leverage architectural improvements suggested by this fictional assessment's findings. TMG Security further recommends that NovaSphere Digital Technologies retest all Critical and High findings following remediation, per the methodology in Section 77, before considering this fictional assessment's core concerns resolved. A follow-on assessment cycle is recommended within twelve months.

THIS IS A FICTIONAL SAMPLE ASSESSMENT.

79 CONTACT & DISCLAIMER

Cybersecurity | *SOC* | *VAPT* | *GRC* | *MDR* | *Compliance Solutions* | *Corporate Trainings*
Web: www.tmgsec.com **Email:** security@tmgsec.com
USA Office: 117 South Lexington Street, STE 100, Harrisonville, MO 64701
Support: support@tmgsec.com | **Phone:** +1 (816) 793-1929
WhatsApp: +1 (480) 680-0342

Final Disclaimer

FICTIONAL SAMPLE REPORT — NOT AN ACTUAL CLIENT iOS PENETRATION TEST

This report was created by TMG Security solely to demonstrate iOS application penetration-testing methodology, technical reporting standards, OWASP MASVS/MASTG/MASWE-aligned evidence presentation, and remediation guidance. NovaSphere Digital Technologies, Inc. and the NovaSphere Pay iOS application (bundle ID com.novasphere.pay.ios) are fictional. No real client environment was assessed. No real iOS application, backend API, or production infrastructure was accessed. No real credentials, API keys, signing certificates, provisioning profiles, personal data, production systems, cloud resources, or customer information were accessed. All Info.plist/entitlements excerpts, decompiled/otool output, Keychain dumps, Console/os_log output, network traffic, and screenshots are synthetic. All vulnerabilities, findings, severity ratings, CVSS scores, and remediation actions are illustrative. No real exploitation occurred. This report is an independent illustrative sample and carries no OWASP affiliation, review, or certification.

80 APPENDIX A — IOS TEST CASE REGISTER

The register below documents all 150 fictional test cases executed during this illustrative assessment, spanning the 24 testing categories presented across the numbered sections of this report, with the applicable MASTG test reference, a result, and a cross-reference to any corresponding finding. Every "Fail" or "Exception Identified" result maps to a finding documented in the detailed findings sections.

Test Case ID	Category	Test Description / Objective	MASTG Reference	Result	Finding
TC-IOS-001	IPA Acquisition	Verify the IPA was acquired through an approved, documented channel (TestFlight / enterprise distribution) and its bundle identifier matches the in-scope target.	N/A	Pass	—
TC-IOS-002	IPA Acquisition	Verify the IPA's code signature is valid and issued by an expected Apple Distribution identity.	MASTG-TEST-0220	Pass	—
TC-IOS-003	IPA Acquisition	Verify the provisioning profile embedded in the IPA is current, correctly scoped, and not expired.	N/A	Observation	IOS-047
TC-IOS-004	IPA Acquisition	Verify a documented lifecycle process exists for signing-certificate rotation and provisioning-profile renewal/revocation.	N/A	Observation	IOS-047
TC-IOS-005	Info.plist / Entitlements	Verify ITSAppUsesNonExemptEncryption accurately reflects the app's cryptography usage.	N/A	Pass	—
TC-IOS-006	Info.plist / Entitlements	Verify UIBackgroundModes are limited to capabilities the app genuinely requires.	N/A	Pass	—
TC-IOS-007	Info.plist / Entitlements	Verify every usage-description key maps to an active, currently shipping feature.	MASTG-TEST-0360	Fail	IOS-022
TC-IOS-008	Info.plist / Entitlements	Verify no stale usage-description keys remain from removed or never-shipped features.	MASTG-TEST-0360	Fail	IOS-028
TC-IOS-009	Info.plist / Entitlements	Verify the keychain-access-groups entitlement follows least privilege across the main app and every extension target.	N/A	Fail	IOS-011
TC-IOS-010	Info.plist / Entitlements	Verify the application-groups entitlement scope matches the minimum data each App Group member actually requires.	MASTG-TEST-0388	Fail	IOS-010
TC-IOS-011	Info.plist / Entitlements	Verify the associated-domains entitlement correctly declares applinks: for every Universal Link path in use.	MASTG-TEST-0395	Fail	IOS-004
TC-IOS-012	Info.plist / Entitlements	Verify the aps-environment entitlement matches the intended push delivery environment (production vs. development).	N/A	Fail	IOS-019
TC-IOS-013	Info.plist / Entitlements	Verify com.apple.developer.default-data-protection is set to an appropriate Data Protection class for the app's sensitivity profile.	MASTG-TEST-0299	Fail	IOS-001
TC-IOS-014	Info.plist / Entitlements	Verify the total count of registered URL schemes and App Extension targets is periodically reviewed for surface-area minimization.	MASTG-TEST-0371	Fail	IOS-032
TC-IOS-015	Info.plist / Entitlements	Verify UIFileSharingEnabled and LSSupportsOpeningDocumentsInPlace are scoped to only the folders intended for file sharing.	N/A	Fail	IOS-016
TC-IOS-016	Binary Analysis	Verify no hardcoded API keys, tokens, or credential material are present in the Mach-O __TEXT.__cstring section.	MASTG-TEST-0213	Fail	IOS-009
TC-IOS-017	Binary Analysis	Verify Position Independent Executable (PIE) is enabled on the release binary.	MASTG-TEST-0228	Pass	—
TC-IOS-018	Binary Analysis	Verify stack canaries are enabled on the release binary.	MASTG-TEST-0229	Fail	IOS-035
TC-IOS-019	Binary Analysis	Verify a CI check exists to automatically verify PIE and stack-canary protections on every release build.	MASTG-TEST-0229	Fail	IOS-035
TC-IOS-020	Binary Analysis	Verify the symbol table does not unnecessarily expose sensitive business-logic class/method names.	N/A	Observation	IOS-039
TC-IOS-021	Binary Analysis	Verify debugging symbols are stripped from the distribution build.	N/A	Pass	—
TC-IOS-022	Binary Analysis	Verify embedded third-party framework binaries match their declared, currently governed SDK versions.	MASTG-TEST-0273	Fail	IOS-031
TC-IOS-023	Static Analysis	Verify deprecated networking/UI APIs with superseded secure alternatives are tracked for removal.	N/A	Fail	IOS-029
TC-IOS-024	Static Analysis	Verify third-party dependencies are scanned for known vulnerabilities on a defined cadence.	MASTG-TEST-0273	Observation	IOS-046
TC-IOS-025	Static Analysis	Verify CI includes an automated security-regression suite covering previously remediated findings.	N/A	Fail	IOS-033
TC-IOS-026	Static Analysis	Verify a documented iOS secure-coding guideline is maintained for the mobile engineering team.	N/A	Fail	IOS-034
TC-IOS-027	Static Analysis	Verify a mandatory pre-release security checklist gate is defined for mobile releases.	N/A	Observation	IOS-044
TC-IOS-028	Static Analysis	Verify malformed or unexpected API responses are handled without leaking internal implementation detail.	MASTG-TEST-0358	Fail	IOS-025
TC-IOS-029	Static Analysis	Verify build-time configuration values are sourced through a centralized secrets-management process rather than ad hoc constants.	N/A	Observation	IOS-040
TC-IOS-030	Static Analysis	Verify Swift/Objective-C static-analysis tooling flags logging calls referencing token/credential-named variables.	MASTG-TEST-0358	Fail	IOS-008
TC-IOS-031	Keychain	Verify session and refresh tokens are stored exclusively in the Keychain, never in NSUserDefaults or a plist.	MASTG-TEST-0300	Fail	IOS-001
TC-IOS-032	Keychain	Verify biometric-gated Keychain items use a device-locked accessibility class combined with a biometry-bound SecAccessControl.	MASTG-TEST-0266	Fail	IOS-002
TC-IOS-033	Keychain	Verify no Keychain item uses kSecAttrAccessibleAlways.	N/A	Fail	IOS-002
TC-IOS-034	Keychain	Verify high-sensitivity session material is provisioned into a Keychain access group scoped to the main app target only.	N/A	Fail	IOS-011
TC-IOS-035	Keychain	Verify Keychain items are excluded from unencrypted local device backups where appropriate.	MASTG-TEST-0298	Pass	—
TC-IOS-036	Keychain	Verify Keychain item access-control policies are validated on a jailbroken fictional test device.	N/A	Fail	IOS-002
TC-IOS-037	Keychain	Verify cryptographic keys backing sensitive operations are Keychain- or Secure-Enclave-resident rather than stored outside the platform keystore.	MASTG-TEST-0213	Pass	—

Test Case ID	Category	Test Description / Objective	MASTG Reference	Result	Finding
TC-IOS-038	Keychain	Verify Keychain query predicates used by App Extensions are scoped to the minimum required access group.	N/A	Fail	IOS-011
TC-IOS-039	Local Storage	Verify no sensitive values are persisted in NSUserDefaults / standard plist files.	MASTG-TEST-0300	Fail	IOS-001
TC-IOS-040	Local Storage	Verify cached transaction-history and statement data are stored using an appropriate Data Protection class.	MASTG-TEST-0300	Fail	IOS-026
TC-IOS-041	Local Storage	Verify the shared App Group container's cached data is minimized and, where retained, appropriately protected.	MASTG-TEST-0388	Fail	IOS-010
TC-IOS-042	Local Storage	Verify Core Data / object-persistence layers do not retain sensitive fields in an unprotected persistent store.	N/A	Fail	IOS-026
TC-IOS-043	Local Storage	Verify temporary files created during statement export are deleted promptly after use.	N/A	Fail	IOS-016
TC-IOS-044	Local Storage	Verify sensitive data is not retained in memory longer than operationally necessary.	N/A	Pass	—
TC-IOS-045	Local Storage	Verify the iOS keyboard cache does not retain sensitive field input (custom keyboards / third-party keyboard extensions).	MASTG-TEST-0313	Pass	—
TC-IOS-046	Local Storage	Verify a fictional unencrypted local device backup does not yield recoverable sensitive application data.	MASTG-TEST-0298	Fail	IOS-001
TC-IOS-047	Local Storage	Verify document-interaction-exposed folders contain only the minimum files necessary for sharing.	N/A	Fail	IOS-016
TC-IOS-048	Cryptography	Verify all sensitive-data encryption uses current, strong, non-deprecated algorithms via CryptoKit / Security framework primitives.	N/A	Pass	—
TC-IOS-049	Cryptography	Verify cryptographic keys are generated with appropriate key sizes for their algorithm.	MASTG-TEST-0209	Pass	—
TC-IOS-050	Cryptography	Verify no custom/home-grown cryptographic implementation is used in place of CryptoKit/Security primitives.	N/A	Pass	—
TC-IOS-051	Cryptography	Verify security-relevant random values (nonces, device-binding secrets) are generated using SecRandomCopyBytes rather than an insecure PRNG.	MASTG-TEST-0311	Fail	IOS-024
TC-IOS-052	Cryptography	Verify no hardcoded cryptographic key material is present anywhere in the compiled application.	MASTG-TEST-0213	Fail	IOS-009
TC-IOS-053	Cryptography	Verify cryptographic key material is never written to os_log under any build configuration.	MASTG-TEST-0296	Fail	IOS-008
TC-IOS-054	Authentication	Verify login rejects invalid credentials with a generic, non-enumerable error.	N/A	Pass	—
TC-IOS-055	Authentication	Verify MFA is enforced for all standard login flows.	N/A	Pass	—
TC-IOS-056	Authentication	Verify MFA is enforced consistently across the account-recovery flow.	N/A	Observation	IOS-036
TC-IOS-057	Authentication	Verify the LAContext biometric-unlock completion handler cannot be bypassed via runtime instrumentation.	MASTG-TEST-0267	Fail	IOS-013
TC-IOS-058	Authentication	Verify biometric authentication is bound to a Keychain/Secure-Enclave-backed credential rather than an app-level boolean flag.	MASTG-TEST-0266	Fail	IOS-013
TC-IOS-059	Authentication	Verify refresh-token rotation invalidates the prior refresh token on use.	N/A	Fail	IOS-014
TC-IOS-060	Authentication	Verify logout triggers server-side token revocation, not only local Keychain-state clearing.	N/A	Fail	IOS-014
TC-IOS-061	Authentication	Verify session idle-timeout enforcement is reasonable for a financial application and independently checked server-side.	N/A	Fail	IOS-020
TC-IOS-062	Authorization	Verify the transfer-confirmation endpoint independently validates that sourceAccountId belongs to the authenticated session.	N/A	Fail	IOS-003
TC-IOS-063	Authorization	Verify object-level authorization is consistent across every account-scoped mobile-API resource type.	N/A	Pass	—
TC-IOS-064	Authorization	Verify high-value transfers are considered for server-side step-up biometric re-authentication.	N/A	Observation	IOS-041
TC-IOS-065	Biometric	Verify Face ID / Touch ID app-unlock cannot be bypassed on a jailbroken fictional test device via completion-handler hooking.	MASTG-TEST-0267	Fail	IOS-013
TC-IOS-066	Biometric	Verify the fallback-to-passcode path does not weaken the effective authentication strength below the intended policy.	MASTG-TEST-0268	Pass	—
TC-IOS-067	Biometric	Verify LAContext evaluatePolicy is invoked with an appropriate LAPolicy value for the sensitivity of the gated action.	MASTG-TEST-0266	Pass	—
TC-IOS-068	Biometric	Verify a second, transaction-scoped biometric prompt is evaluated for high-value transfers beyond app-unlock.	N/A	Observation	IOS-041
TC-IOS-069	Biometric	Verify the biometric-vault Keychain item's accessibility class and access-control policy match the intended local-authentication gate.	MASTG-TEST-0266	Fail	IOS-002
TC-IOS-070	Network	Verify all mobile-API traffic is transmitted exclusively over TLS.	MASTG-TEST-0322	Pass	—
TC-IOS-071	Network	Verify the minimum accepted TLS protocol version is 1.2 or higher on every first-party domain.	MASTG-TEST-0342	Fail	IOS-021
TC-IOS-072	Network	Verify no NSExceptionDomains entry permits a TLS version below 1.2.	MASTG-TEST-0342	Fail	IOS-021
TC-IOS-073	Network	Verify ATS configuration applies domain-level granularity distinguishing payments-critical domains from lower-sensitivity third-party SDK domains.	MASTG-TEST-0396	Fail	IOS-027
TC-IOS-074	Network	Verify certificate/public-key pinning is implemented for api.novasphere-example.com and auth.novasphere-example.com.	MASTG-TEST-0385	Pass	—
TC-IOS-075	Network	Verify custom URLSessionDelegate trust-evaluation implementations enforce SecTrustEvaluateWithError as a hard gate.	MASTG-TEST-0396	Fail	IOS-007
TC-IOS-076	Network	Verify custom WKNavigationDelegate implementations do not bypass certificate validation.	MASTG-TEST-0397	Pass	—
TC-IOS-077	Network	Verify the app resists a controlled interception-proxy attempt on the primary payments API channel.	MASTG-TEST-0385	Pass	IOS-037
TC-IOS-078	Network	Verify the primary API domains' TLS negotiation uses a strong cipher suite and protocol version.	MASTG-TEST-0342	Pass	IOS-037
TC-IOS-079	Network	Verify no request/response header logging captures sensitive Authorization or MFA values.	MASTG-TEST-0296	Fail	IOS-008
TC-IOS-080	Network	Verify the legacy statement-export domain's ATS exception is removed once the backend supports TLS 1.2+.	MASTG-TEST-0342	Fail	IOS-021
TC-IOS-081	WKWebView	Verify WKScriptMessageHandler bridge methods do not expose live session-token material to WebView-side JavaScript.	MASTG-TEST-0376	Fail	IOS-006
TC-IOS-082	WKWebView	Verify WKScriptMessageHandler implementations validate message.frameInfo.securityOrigin before responding.	MASTG-TEST-0376	Fail	IOS-006

Test Case ID	Category	Test Description / Objective	MASTG Reference	Result	Finding
TC-IOS-083	WKWebView	Verify WKWebView navigation is restricted to an allow-listed first-party domain set.	N/A	Fail	IOS-006
TC-IOS-084	WKWebView	Verify mixed-content loading is disabled in all WKWebView instances.	N/A	Pass	—
TC-IOS-085	WKWebView	Verify WKWebView JavaScript execution is disabled where not strictly required.	N/A	Pass	—
TC-IOS-086	Deep Links	Verify all custom-scheme deep links route through in-app confirmation before committing a sensitive account change.	MASTG-TEST-0370	Fail	IOS-004
TC-IOS-087	Deep Links	Verify Universal Link handlers validate the apple-app-site-association association rather than accepting an unverified custom scheme for sensitive flows.	MASTG-TEST-0395	Fail	IOS-004
TC-IOS-088	Deep Links	Verify the OAuth account-linking callback migrates to a Universal Link with server-side state/nonce validation.	MASTG-TEST-0371	Fail	IOS-005
TC-IOS-089	Deep Links	Verify custom URL scheme handlers validate the calling source where the platform allows.	MASTG-TEST-0371	Fail	IOS-005
TC-IOS-090	Deep Links	Verify Universal Link handler parameters are treated as untrusted input and validated server-side.	MASTG-TEST-0395	Fail	IOS-004
TC-IOS-091	Deep Links	Verify the link-payee Universal Link path no longer supports a silent auto-confirm parameter.	MASTG-TEST-0395	Fail	IOS-004
TC-IOS-092	Deep Links	Verify registered custom URL schemes and Universal Link paths are periodically reviewed for surface-area minimization.	N/A	Fail	IOS-032
TC-IOS-093	Deep Links	Verify Siri Shortcuts / App Intents exposure is scoped to non-sensitive actions only.	N/A	Pass	IOS-045
TC-IOS-094	Deep Links	Verify no App Intent directly performs a state-changing financial action without in-app confirmation.	N/A	Pass	IOS-045
TC-IOS-095	App Extensions	Verify each App Extension target receives only the minimum data needed for its specific task.	N/A	Fail	IOS-017
TC-IOS-096	App Extensions	Verify the share extension does not read the main app's full cached transaction-history file.	N/A	Fail	IOS-017
TC-IOS-097	App Extensions	Verify the widget extension's shared App Group cache excludes reversible financial identifiers (e.g., full/partial card numbers) beyond minimized display data.	MASTG-TEST-0388	Fail	IOS-010
TC-IOS-098	App Extensions	Verify the Notification Service Extension redacts sensitive transaction detail before lock-screen display.	N/A	Fail	IOS-019
TC-IOS-099	App Extensions	Verify the Notification Content Extension does not access Keychain or App Group data beyond notification rendering.	N/A	Pass	—
TC-IOS-100	App Extensions	Verify App Group container contents are reviewed on a jailbroken fictional test device for unintended sensitive-data exposure.	MASTG-TEST-0388	Fail	IOS-010
TC-IOS-101	App Extensions	Verify the total count of App Extension targets is periodically reviewed for surface-area minimization.	N/A	Fail	IOS-032
TC-IOS-102	Pasteboard	Verify sensitive values copied to UIPasteboard set an expirationDate.	MASTG-TEST-0276	Fail	IOS-012
TC-IOS-103	Pasteboard	Verify sensitive values copied to UIPasteboard set localOnly to prevent Universal Clipboard propagation.	MASTG-TEST-0276	Fail	IOS-012
TC-IOS-104	Pasteboard	Verify UIPasteboard.setItems(_:options:) is used in place of the bare .string setter for sensitive copy actions.	MASTG-TEST-0277	Fail	IOS-012
TC-IOS-105	Pasteboard	Verify no other in-app copy action writes sensitive data to the general pasteboard without expiration.	MASTG-TEST-0276	Pass	—
TC-IOS-106	Document Interaction	Verify UIFileSharingEnabled does not expose the entire Documents directory to Files app / Finder sharing.	N/A	Fail	IOS-016
TC-IOS-107	Document Interaction	Verify cached statement export PDFs are deleted promptly after the user completes the share/export action.	N/A	Fail	IOS-016
TC-IOS-108	Document Interaction	Verify LSSupportsOpeningDocumentsInPlace scope matches a documented, narrow export use case.	N/A	Fail	IOS-016
TC-IOS-109	Document Interaction	Verify UIActivityViewController share-sheet integrations do not leak unintended file contents.	N/A	Pass	—
TC-IOS-110	Push Notifications	Verify lock-screen notification content redacts sensitive transaction detail by default.	MASTG-TEST-0346	Fail	IOS-019
TC-IOS-111	Push Notifications	Verify a UNNotificationServiceExtension or summaryArgument is used to reveal full detail only when appropriate.	MASTG-TEST-0346	Fail	IOS-019
TC-IOS-112	Push Notifications	Verify the aps-environment entitlement matches the intended push delivery environment.	N/A	Fail	IOS-019
TC-IOS-113	Push Notifications	Verify push-token registration requests are authenticated and scoped to the registering session.	N/A	Pass	—
TC-IOS-114	Screenshot / Snapshot	Verify a privacy overlay is installed before backgrounding on the account-balance and card-detail screens.	MASTG-TEST-0290	Fail	IOS-015
TC-IOS-115	Screenshot / Snapshot	Verify the App Switcher snapshot does not display unobscured sensitive account or card detail.	MASTG-TEST-0290	Fail	IOS-015
TC-IOS-116	Screenshot / Snapshot	Verify the on-disk background snapshot cache file does not retain unobscured sensitive content.	MASTG-TEST-0290	Fail	IOS-015
TC-IOS-117	Logging	Verify os_log calls in the networking and authentication modules do not log bearer tokens or MFA codes.	MASTG-TEST-0296	Fail	IOS-008
TC-IOS-118	Logging	Verify os_log {private} formatting is applied by default to any variable sourced from token/credential storage.	MASTG-TEST-0296	Fail	IOS-008
TC-IOS-119	Logging	Verify non-sensitive diagnostic modules apply consistent logging-hygiene standards in release builds.	MASTG-TEST-0297	Fail	IOS-030
TC-IOS-120	Logging	Verify verbose QA-build logging configuration is not present in the distribution build.	MASTG-TEST-0358	Fail	IOS-008
TC-IOS-121	Logging	Verify malformed-request error handling does not surface internal implementation detail via a debug alert path.	MASTG-TEST-0358	Fail	IOS-025
TC-IOS-122	Privacy	Verify only permissions backing an active, shipping feature are requested.	MASTG-TEST-0360	Fail	IOS-022
TC-IOS-123	Privacy	Verify unused usage-description keys are removed rather than retained from discontinued features.	MASTG-TEST-0360	Fail	IOS-028
TC-IOS-124	Privacy	Verify permission requests are made just-in-time at point of feature use rather than in a first-launch batch.	MASTG-TEST-0361	Fail	IOS-022
TC-IOS-125	Privacy	Verify the AnalyticsKit tracking identifier resets appropriately on app deletion/reinstall rather than persisting via the Keychain.	N/A	Fail	IOS-023

Test Case ID	Category	Test Description / Objective	MASTG Reference	Result	Finding
TC-IOS-126	Privacy	Verify crash-reporting breadcrumbs are reviewed for incidental PII capture.	N/A	Observation	IOS-043
TC-IOS-127	Privacy	Verify users can access an in-app data-usage/privacy disclosure consistent with MASVS-PRIVACY-3.	N/A	Pass	—
TC-IOS-128	Privacy	Verify users have a mechanism to request account/data deletion consistent with MASVS-PRIVACY-4.	N/A	Pass	—
TC-IOS-129	Third-Party SDKs	Verify every embedded third-party SDK is documented with owner, purpose, and data-access scope in a maintained inventory.	MASTG-TEST-0273	Observation	IOS-042
TC-IOS-130	Third-Party SDKs	Verify all third-party SDKs, including lower-usage ones, have a documented review cadence and version-pinning policy.	MASTG-TEST-0273	Fail	IOS-031
TC-IOS-131	Third-Party SDKs	Verify the mapping SDK API key is not compiled as a plaintext binary string literal.	MASTG-TEST-0213	Fail	IOS-009
TC-IOS-132	Third-Party SDKs	Verify dependency-vulnerability scanning runs on a defined recurring cadence rather than only at build time.	MASTG-TEST-0273	Observation	IOS-046
TC-IOS-133	Third-Party SDKs	Verify third-party SDKs do not silently expand their data collection beyond documented scope.	MASTG-TEST-0273	Pass	—
TC-IOS-134	Jailbreak Detection	Verify jailbreak detection combines multiple independent indicators rather than a single Cydia-path/fork() check.	MASTG-TEST-0240	Fail	IOS-018
TC-IOS-135	Jailbreak Detection	Verify jailbreak detection is not defeated by a commodity jailbreak-hiding tweak in fictional testing.	MASTG-TEST-0240	Fail	IOS-018
TC-IOS-136	Jailbreak Detection	Verify jailbreak-detection logic is exercised at runtime, not only referenced statically in code.	MASTG-TEST-0241	Fail	IOS-018
TC-IOS-137	Jailbreak Detection	Verify Apple DeviceCheck or App Attest is evaluated as a server-verifiable integrity signal.	N/A	Observation	IOS-038
TC-IOS-138	Jailbreak Detection	Verify the app degrades gracefully (rather than silently) when a jailbreak signal is detected.	N/A	Pass	—
TC-IOS-139	Jailbreak Detection	Verify reverse-engineering tooling (common hooking frameworks) is considered in the jailbreak-risk-scoring model.	N/A	Pass	—
TC-IOS-140	Anti-Tampering	Verify binary obfuscation is evaluated for sensitive payments business-logic modules.	MASTG-TEST-0391	Observation	IOS-039
TC-IOS-141	Anti-Tampering	Verify signing-certificate and provisioning-profile lifecycle procedures are formally documented.	N/A	Observation	IOS-047
TC-IOS-142	Anti-Tampering	Verify a secure-SDLC checkpoint gate is established for mobile release sign-off.	N/A	Observation	IOS-044
TC-IOS-143	API / Session	Verify the transfer-confirmation endpoint rejects a client-supplied sourceAccountId not owned by the authenticated session.	N/A	Fail	IOS-003
TC-IOS-144	API / Session	Verify the support-WebView session-token endpoint issues narrowly scoped, short-lived tokens.	MASTG-TEST-0376	Fail	IOS-006
TC-IOS-145	API / Session	Verify the telemetry ingestion endpoint validates device-token authenticity.	N/A	Fail	IOS-023
TC-IOS-146	API / Session	Verify token replay after logout is rejected by the backend once revocation is implemented.	N/A	Fail	IOS-014
TC-IOS-147	API / Session	Verify session idle-timeout has a corresponding server-side session-age check independent of the client-side timer.	N/A	Fail	IOS-020
TC-IOS-148	API / Session	Verify mobile-API error responses use a sanitized, generic error contract.	MASTG-TEST-0358	Fail	IOS-025
TC-IOS-149	API / Session	Verify the biometric-challenge endpoint is integrated into high-value transfer confirmation as a step-up control.	N/A	Observation	IOS-041
TC-IOS-150	API / Session	Verify MFA enforcement on account recovery has no client-side bypass path across all tested scenarios.	N/A	Pass	IOS-036

81 APPENDIX B — IPA METADATA REGISTER

This appendix documents the complete fictional IPA metadata, third-party SDK inventory, native library footprint, and suspicious-string findings collected during static analysis of NovaSpherePay-6.4.2.ipa.

Core IPA Metadata

Field	Value
IPA File Name	NovaSpherePay-6.4.2.ipa
Bundle Identifier	com.novasphere.pay.ios
Bundle Short Version String / Bundle Version	6.4.2 (build 6402)
Minimum iOS Deployment Target	16.0
Target / Compiled SDK	iOS 18 SDK
Supported Architecture	arm64 (arm64e not observed)
Primary Language / Runtime	Swift 5.x, with a small number of legacy Objective-C bridging headers
Code Signing Scheme	Apple Distribution certificate + App Store provisioning profile (fictional production identity; no real fingerprint disclosed)
Provisioning Profile Type	App Store distribution provisioning profile (fictional)
Entitlements	See Appendix D — Entitlements Register
Encryption Export Compliance (ITSAppUsesNonExemptEncryption)	NO — standard HTTPS/TLS exempt encryption only
Symbol Stripping / Obfuscation	Release build strips debug symbols; no name-mangling or control-flow obfuscation applied — see IOS-039
Debuggable Entitlement (get-task-allow)	Not present in the distribution build (confirmed via codesign -d --entitlements)
Build Tool	Xcode 16.x (fictional build configuration), Swift Package Manager for dependency resolution

Third-Party SDK / Framework Inventory

Framework	Vendor (Fictional)	Purpose	Data Access / Notes
UIKit	Apple System Framework	Primary UI framework	No embedded copy — system-provided
SwiftUI	Apple System Framework	Widget extension and select modern screens	No embedded copy — system-provided
Foundation	Apple System Framework	Core runtime services	No embedded copy — system-provided
Security	Apple System Framework	Keychain Services, SecTrust APIs	No embedded copy — system-provided; see IOS-001, IOS-002, IOS-011
LocalAuthentication	Apple System Framework	Face ID / Touch ID app-unlock gate	No embedded copy — system-provided; see IOS-013
WebKit	Apple System Framework	Support/help WKWebView	No embedded copy — system-provided; see IOS-006
CryptoKit	Apple System Framework	Device-binding nonce and local hashing operations	No embedded copy — system-provided; see IOS-024
UserNotifications	Apple System Framework	Transaction/security alert push notifications	No embedded copy — system-provided; see IOS-019
AuthenticationServices	Apple System Framework	Third-party bank-account aggregation OAuth presentation (ASWebAuthenticationSession)	No embedded copy — system-provided; see IOS-005

Framework	Vendor (Fictional)	Purpose	Data Access / Notes
MapSDKKit	Fictional Third-Party Vendor A	Branch/ATM-locator map rendering	Embedded framework — see IOS-009 (hardcoded API key)
AnalyticsKit	Fictional Third-Party Vendor B	Usage analytics, funnel tracking	Embedded framework — see IOS-023, Appendix M
CrashReportKit	Fictional Third-Party Vendor C	Crash and non-fatal error reporting	Embedded framework — see IOS-043, Appendix M
PartnerLinkKit	Fictional Third-Party Vendor E	Bank-account aggregation OAuth account-linking flow	Embedded framework — see IOS-005, Appendix M
ImageCacheKit	Fictional Third-Party Vendor D	Remote avatar/promo image loading and caching	Embedded framework — see IOS-031, Appendix M
DocExportKit	Fictional Third-Party Vendor G (legacy)	Legacy statement-export backend client	Embedded framework — see IOS-007, IOS-021, Appendix M

Native Library / Mach-O Footprint

Library / Binary	Architecture	Notes
NovaSpherePay	Main Mach-O executable (arm64)	Primary application binary.
libnovasphere-crypto.dylib	arm64	Fictional native helper wrapping select CryptoKit/Security operations.
MapSDKKit.framework/MapSDKKit	arm64	Third-party mapping SDK embedded framework binary.
AnalyticsKit.framework/AnalyticsKit	arm64	Third-party analytics SDK embedded framework binary.
CrashReportKit.framework/ CrashReportKit	arm64	Third-party crash-reporting SDK embedded framework binary.
PartnerLinkKit.framework/ PartnerLinkKit	arm64	Third-party bank-aggregation OAuth SDK embedded framework binary.
ImageCacheKit.framework/ ImageCacheKit	arm64	Third-party image loading/caching SDK embedded framework binary.
DocExportKit.framework/DocExportKit	arm64	Legacy third-party statement-export client embedded framework binary.

Suspicious / Notable Static Strings

String (Fictional)	Context
https://api.novasphere-example.com/v6/	Production mobile-API base URL (fictional, non-resolving).
https://auth.novasphere-example.com/oauth/	Production auth/OAuth base URL (fictional, non-resolving).
https://docs.novasphere-example.com/export/	Legacy statement-export document backend URL — see IOS-007, IOS-021 (fictional, non-resolving).
https://staging-api.novasphere-example.com/v6/	Fictional staging endpoint reference retained in the release binary's string table.
MAPS_API_KEY=fictional_maps_sdk_key_8a41cf209b	Hardcoded mapping SDK key — see IOS-009.
novasphere://oauth-callback	Custom URL scheme OAuth callback — see IOS-005.
// TODO: remove QA verbose logging flag before App Store submission	Fictional residual developer comment referencing the verbose-logging configuration tracked in IOS-008.
FICTIONAL_DEBUG_MENU_ENABLED=0	Fictional residual debug-menu compile flag string, confirmed disabled in the release configuration.

82 APPENDIX C — INFO.PLIST REGISTER

The register below documents all 15 fictional Info.plist keys reviewed during static analysis of NovaSpherePay-6.4.2.ipa, with each key's declared value, review notes, and any corresponding finding.

Info.plist Key	Declared Value	Notes	Finding
CFBundleIdentifier	com.novasphere.pay.ios	Application bundle identifier.	—
CFBundleShortVersionString	6.4.2	Marketing version string.	—
CFBundleVersion	6402	Build number.	—
UIRequiredDeviceCapabilities	arm64	Standard 64-bit device capability requirement.	—
NSAppTransportSecurity	Base policy: TLS 1.2+, no arbitrary loads. NSExceptionDomains: docs.novasphere-example.com (NSExceptionMinimumTLSVersion: TLSv1.1)	Legacy statement-export domain exception weakens the TLS floor for one domain; no per-domain granularity for payments-critical domains.	IOS-021, IOS-027
LSApplicationQueriesSchemes	fictionalbankaggregator, fictionalwallet, mapsapp	External app schemes NovaSphere Pay is permitted to query with canOpenURL(_:).	—
CFBundleURLTypes	novasphere, novasphere-support, novasphere-widget, novasphere-legacy, novasphere-promo (see Appendix E for full scheme/handler detail)	Six custom URL scheme entries registered; several correspond to narrow or legacy-only functionality.	IOS-005, IOS-032
NSFaceIDUsageDescription	"NovaSphere Pay uses Face ID to securely unlock your account."	Backs the biometric app-unlock gate.	IOS-013
NSCameraUsageDescription	"NovaSphere Pay uses the camera for check deposit."	Declared, but static/dynamic review found no current UI flow that requests camera access.	IOS-028
NSCalendarsUsageDescription	"NovaSphere Pay would like to access your calendar to schedule payment reminders."	Declared, but no current UI flow requests calendar access.	IOS-028
NSContactsUsageDescription	"NovaSphere Pay uses your contacts to help you pay friends."	Requested at first launch; no shipping feature currently references Contacts.	IOS-022
NSMicrophoneUsageDescription	"NovaSphere Pay uses the microphone for voice memo support."	Requested at first launch; no shipping feature currently references the microphone.	IOS-022
NSLocationWhenInUseUsageDescription	"NovaSphere Pay uses your location to find nearby branches and ATMs."	Backs the branch/ATM-locator feature (MapSDKKit); scoped, contextual usage.	—
UIBackgroundModes	remote-notification, fetch	Supports push-triggered background refresh and periodic balance sync.	—
ITSAppUsesNonExemptEncryption	NO	Standard HTTPS/TLS only; no non-exempt (proprietary) encryption shipped in the binary.	—

83 APPENDIX D — ENTITLEMENTS REGISTER

The register below documents all 6 fictional entitlements declared in the embedded provisioning profile / entitlements.plist for NovaSpherePay-6.4.2.ipa, with declared value, notes, and any corresponding finding.

Entitlement	Declared Value	Notes	Finding
keychain-access-groups	$(AppIdentifierPrefix)com.novasphere.pay.ios.shared	Shared Keychain access group between the main app and the Home Screen widget extension.	IOS-002, IOS-011
com.apple.developer.associated-domains	applinks:app.novasphere-example.com	Universal Link verification domain for the beneficiary-linking, statement, support, and promo flows.	IOS-004, IOS-032
com.apple.security.application-groups	group.com.novasphere.pay.ios.shared	Shared container between the main app, the widget extension, and the share extension.	IOS-010, IOS-017
aps-environment	production	Enables APNs push delivery for transaction and security alert notifications.	IOS-019
com.apple.developer.authentication-services.autofill-credential-provider	Not declared	The app does not implement a credential-provider extension; login uses the in-app form only.	—
com.apple.developer.default-data-protection	NSFileProtectionCompleteUntilFirstUserAuthentication	Default Data Protection class applied to the app's on-disk container; several specific files do not raise this to a stricter class — see storage findings.	IOS-001, IOS-026

84 APPENDIX E — URL SCHEME / UNIVERSAL LINK REGISTER

The register below documents all 11 fictional custom URL scheme entries and Universal Link routes identified in NovaSpherePay-6.4.2.ipa, with type, handler, review notes, and any corresponding finding.

Scheme / Link	Type	Handler	Notes	Finding
novasphere://oauth-callback	Custom URL Scheme	application(_:open:options:) — PartnerLinkKit OAuth callback handler	Unreserved scheme; no origin verification and no observed state/nonce validation on the callback.	IOS-005
novasphere://add-payee	Custom URL Scheme (legacy)	application(_:open:options:) — legacy PayeeLinkHandler alias predating the Universal Link migration	Retained for backward compatibility; narrows to the same underlying handler as the /link-payee Universal Link.	IOS-032
novasphere-support://	Custom URL Scheme (legacy, narrow usage)	SupportDeepLinkHandler — opens the in-app support WebView	Narrow current usage; candidate for retirement/consolidation.	IOS-032
novasphere-widget://	Custom URL Scheme (internal)	WidgetActionHandler — Home Screen widget quick-action tap-through	Internal widget-to-app link; not intended for third-party use.	IOS-032
novasphere-legacy://	Custom URL Scheme (deprecated)	LegacyRouteHandler — pre-6.0 scheme retained for backward compatibility	Deprecated; no current first-party callers identified.	IOS-032
novasphere-promo://	Custom URL Scheme (marketing)	PromoDeepLinkHandler — marketing/promo campaign landing	Narrow, campaign-scoped usage.	IOS-032
https://app.novasphere-example.com/link-payee	Universal Link	NSUserActivity continuation — PayeeLinkHandler (auto-confirm path)	Silently commits a new payee with no in-app confirmation when auto=1 is present.	IOS-004
https://app.novasphere-example.com/statement	Universal Link	NSUserActivity continuation — StatementDetailHandler	Requires an active session; opens statement detail within the app, no state-changing action.	—
https://app.novasphere-example.com/support	Universal Link	NSUserActivity continuation — SupportDeepLinkHandler	Opens a specific support article in the in-app WKWebView.	IOS-006
https://app.novasphere-example.com/promo	Universal Link	NSUserActivity continuation — PromoDeepLinkHandler	Opens a marketing landing screen; no account-data action performed.	—
https://app.novasphere-example.com/oauth-callback	Universal Link (planned)	Not yet implemented — recommended migration target for the OAuth account-linking callback	Recommended replacement for the novasphere://oauth-callback custom-scheme flow.	IOS-005

85 APPENDIX F — APP EXTENSION REGISTER

The register below documents all 4 fictional app extension targets bundled in NovaSpherePay-6.4.2.ipa, their extension point, data/App Group access, review notes, and any corresponding finding.

Extension	Extension Point	Data / App Group Access	Notes	Finding
NovaSpherePay Widget	Home Screen Widget Extension (WidgetKit)	group.com.novasphere.pay.ios.shared — cached account balance, last-four card digits	Shares the same Keychain access group as the main app's refresh token (IOS-011); shared App Group cache is unencrypted (IOS-010).	IOS-010, IOS-011
NovaSpherePay Share Extension	Action / Share Extension	group.com.novasphere.pay.ios.shared — full cached transaction-history file (not scoped to the shared item)	Reads the entire cached transaction-history file rather than a scoped single-transaction payload.	IOS-017
NovaSpherePay Notification Service Extension	Notification Service Extension (UNNotificationServiceExtension)	Push payload delivered by APNs — no App Group data read	Not currently used to redact sensitive transaction detail before lock-screen display.	IOS-019
NovaSpherePay Notification Content Extension	Notification Content Extension	Push payload rendering only — no App Group or Keychain data accessed	No elevated data access identified; scoped strictly to expanded notification UI rendering.	—

86 APPENDIX G — API / ENDPOINT REGISTER

The register below documents all 27 fictional backend API endpoints identified during mobile client traffic analysis of NovaSphere Pay for iOS, with host, method, authentication requirement, business criticality, function, and any corresponding finding.

Host	Method	Path	Auth	Criticality	Function	Finding
auth.novasphere-example.com	POST	/oauth/token	None	Critical	Credential login; issues access and refresh tokens.	—
auth.novasphere-example.com	POST	/oauth/token/refresh	Refresh Token	Critical	Access-token refresh.	—
auth.novasphere-example.com	POST	/oauth/revoke	Session	High	Token revocation on logout.	IOS-014
auth.novasphere-example.com	POST	/oauth/mfa/verify	Partial Session	Critical	MFA one-time-code verification.	—
auth.novasphere-example.com	POST	/oauth/biometric/register	Session	High	Registers the biometric-unlock device-binding key.	IOS-002
auth.novasphere-example.com	POST	/oauth/link/callback	State/Nonce (expected)	Critical	Bank-account aggregation OAuth account-linking callback.	IOS-005
api.novasphere-example.com	GET	/v6/mobile/accounts	Session	High	Account list for the session owner.	—
api.novasphere-example.com	GET	/v6/mobile/accounts/{accountId}	Session	High	Account detail retrieval.	—
api.novasphere-example.com	GET	/v6/mobile/accounts/{accountId}/statements	Session	High	Statement listing retrieval.	—
api.novasphere-example.com	GET	/v6/mobile/statements/{statementId}/export	Session	High	PDF statement export generation (legacy DocExportKit backend).	IOS-007, IOS-016, IOS-021
api.novasphere-example.com	POST	/v6/mobile/transfers	Session	Critical	Funds-transfer initiation.	IOS-003
api.novasphere-example.com	POST	/v6/mobile/transfers/confirm	Session	Critical	Funds-transfer confirmation — object-level authorization gap.	IOS-003
api.novasphere-example.com	GET	/v6/mobile/transfers/{transferId}	Session	High	Transfer status retrieval.	—
api.novasphere-example.com	GET	/v6/mobile/payees	Session	High	Payee list retrieval.	—
api.novasphere-example.com	POST	/v6/mobile/payees	Session	Critical	Payee creation — reachable silently via Universal Link auto-confirm.	IOS-004
api.novasphere-example.com	DELETE	/v6/mobile/payees/{payeeId}	Session	High	Payee removal.	—
api.novasphere-example.com	GET	/v6/mobile/cards	Session	High	Linked card list retrieval.	—
api.novasphere-example.com	GET	/v6/mobile/cards/{cardId}/detail	Session	Critical	Full (client-masked) card detail retrieval.	IOS-015
api.novasphere-example.com	GET	/v6/mobile/profile	Session	Medium	User profile retrieval.	—
api.novasphere-example.com	PATCH	/v6/mobile/profile	Session	Medium	User profile update.	—
api.novasphere-example.com	POST	/v6/mobile/devices/push-token	Session	Low	Device/push-token registration.	IOS-019
api.novasphere-example.com	POST	/v6/mobile/biometric-challenge	Session	High	Server-side biometric step-up challenge/response (not currently invoked for high-value transfers).	IOS-041
api.novasphere-example.com	GET	/v6/mobile/notifications/preferences	Session	Low	Notification preference retrieval.	—
api.novasphere-example.com	GET	/v6/mobile/support/articles	None	Low	Static support-article listing.	—
api.novasphere-example.com	POST	/v6/mobile/support/session-token	Session	Medium	Support-WKWebView session token issuance.	IOS-006
api.novasphere-example.com	POST	/v6/mobile/telemetry/events	Device Token	Low	AnalyticsKit telemetry event ingestion.	IOS-023
api.novasphere-example.com	GET	/v6/mobile/config/feature-flags	None	Low	Feature-flag configuration retrieval.	—

87 APPENDIX H — MASVS COVERAGE MATRIX

The matrix below maps this fictional assessment's coverage against all 24 current OWASP MASVS controls across the eight MASVS-STORAGE, MASVS-CRYPTO, MASVS-AUTH, MASVS-NETWORK, MASVS-PLATFORM, MASVS-CODE, MASVS-RESILIENCE, and MASVS-PRIVACY categories, consistent with the coverage model introduced earlier in this report.

MASVS Control	Control Statement	Tested?	Method	Result	Related Findings
MASVS-STORAGE-1	*The app securely stores sensitive data.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	Weaknesses Identified	IOS-001, IOS-002, IOS-009, IOS-010, IOS-026, IOS-040
MASVS-STORAGE-2	*The app prevents leakage of sensitive data.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	Weaknesses Identified	IOS-008, IOS-030
MASVS-CRYPTO-1	*The app employs current strong cryptography and uses it according to industry best practices.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	Weaknesses Identified	IOS-024
MASVS-CRYPTO-2	*The app performs key management according to industry best practices.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	No Exceptions Identified	—
MASVS-AUTH-1	*The app uses secure authentication and authorization protocols and follows the relevant best practices.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	Weaknesses Identified	IOS-003, IOS-014, IOS-020, IOS-036
MASVS-AUTH-2	*The app performs local authentication securely according to the platform best practices.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	Weaknesses Identified	IOS-013, IOS-041
MASVS-AUTH-3	*The app secures sensitive operations with additional authentication.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	No Exceptions Identified	—
MASVS-NETWORK-1	*The app secures all network traffic according to the current best practices.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	Weaknesses Identified	IOS-021, IOS-027, IOS-037
MASVS-NETWORK-2	*The app performs identity pinning for all remote endpoints under the developer's control.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	Weaknesses Identified	IOS-007
MASVS-PLATFORM-1	*The app uses IPC mechanisms securely.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	Weaknesses Identified	IOS-004, IOS-005, IOS-011, IOS-016, IOS-017, IOS-032, IOS-045
MASVS-PLATFORM-2	*The app uses WebViews securely.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	Weaknesses Identified	IOS-006
MASVS-PLATFORM-3	*The app uses the user interface securely.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	Weaknesses Identified	IOS-012, IOS-015, IOS-019
MASVS-CODE-1	*The app requires an up-to-date platform version.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	Weaknesses Identified	IOS-029, IOS-034, IOS-044, IOS-047
MASVS-CODE-2	*The app has a mechanism for enforcing app updates.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	Weaknesses Identified	IOS-031, IOS-042, IOS-046
MASVS-CODE-3	*The app only uses software components without known vulnerabilities.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	No Exceptions Identified	—
MASVS-CODE-4	*The app validates and sanitizes all untrusted inputs.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	Weaknesses Identified	IOS-025, IOS-033
MASVS-RESILIENCE-1	*The app validates the integrity of the platform.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	Weaknesses Identified	IOS-018
MASVS-RESILIENCE-2	*The app implements anti-tampering mechanisms.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	Weaknesses Identified	IOS-039
MASVS-RESILIENCE-3	*The app implements anti-static analysis mechanisms.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	No Exceptions Identified	—
MASVS-RESILIENCE-4	*The app implements anti-dynamic analysis techniques.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	Weaknesses Identified	IOS-035, IOS-038
MASVS-PRIVACY-1	*The app minimizes access to sensitive data and resources.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	Weaknesses Identified	IOS-022, IOS-028
MASVS-PRIVACY-2	*The app prevents identification of the user.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	Weaknesses Identified	IOS-023, IOS-043
MASVS-PRIVACY-3	*The app is transparent about data collection and usage.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	No Exceptions Identified	—
MASVS-PRIVACY-4	*The app offers user control over their data.*	Yes	Static (IPA/Info.plist/entitlements/Mach-O) + dynamic (jailbroken and non-jailbroken device) analysis per Sections 26-92	No Exceptions Identified	—

MASVS control identifiers and statements above are drawn from current, publicly verifiable OWASP MASVS material. This report carries no OWASP affiliation, review, or certification.

88 APPENDIX I — MASWE / MASTG TRACEABILITY

The table below traces each of the 67 rows in this appendix from MASVS control, through the specific OWASP MASWE weakness and MASTG-TEST applied, to the testing result, corresponding finding, and evidence reference — the full traceability chain applied throughout this assessment.

MASVS Control	MASWE Weakness	MASTG Test	Result	Finding	Evidence
MASVS-STORAGE-1	MASWE-0001 — Sensitive Data Stored Unencrypted in Private Storage	MASTG-TEST-0300 — References to APIs for Storing Unencrypted Data in Private Storage	Exception Identified	IOS-001	EV-IOS-001
MASVS-STORAGE-1	MASWE-0016 — Cryptographic Key Access Not Restricted	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	Exception Identified	IOS-002	EV-IOS-003
MASVS-AUTH-1	MASWE-0018 — Lack of Authentication or Authorization on App Components	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	Exception Identified	IOS-003	EV-IOS-005
MASVS-PLATFORM-1	MASWE-0029 — Insecure Deep Links	MASTG-TEST-0395 — Missing Input Validation in Universal Link Handlers	Exception Identified	IOS-004	EV-IOS-006
MASVS-PLATFORM-1	MASWE-0029 — Insecure Deep Links	MASTG-TEST-0371 — Missing Source Validation in Custom URL Scheme Handlers	Exception Identified	IOS-005	EV-IOS-007
MASVS-PLATFORM-2	MASWE-0034 — WebViews Allow Access to Local Resources with Untrusted Content	MASTG-TEST-0376 — References to Native Bridge APIs in WebViews	Exception Identified	IOS-006	EV-IOS-008
MASVS-NETWORK-2	MASWE-0027 — Insecure Certificate Validation	MASTG-TEST-0396 — References to URLSessionDelegate Bypassing Certificate Validation	Exception Identified	IOS-007	EV-IOS-009
MASVS-STORAGE-2	MASWE-0005 — Insertion of Sensitive Data into Logs	MASTG-TEST-0296 — Sensitive Data Exposure in Logs	Exception Identified	IOS-008	EV-IOS-010
MASVS-STORAGE-1	MASWE-0004 — Sensitive Data Hardcoded in the App Package	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	Exception Identified	IOS-009	EV-IOS-011
MASVS-STORAGE-1	MASWE-0001 — Sensitive Data Stored Unencrypted in Private Storage	MASTG-TEST-0388 — References to Sensitive Data Stored Unprotected in Shared App Group Containers	Exception Identified	IOS-010	EV-IOS-012
MASVS-PLATFORM-1	MASWE-0016 — Cryptographic Key Access Not Restricted	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	Exception Identified	IOS-011	EV-IOS-013
MASVS-AUTH-2	MASWE-0020 — Local Authentication Can Be Bypassed	MASTG-TEST-0266 — References to APIs for Event-Bound Biometric Authentication	Exception Identified	IOS-013	EV-IOS-014
MASVS-AUTH-1	MASWE-0024 — Sensitive Data Accessible After Session Termination	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	Exception Identified	IOS-014	EV-IOS-015
MASVS-PLATFORM-3	MASWE-0036 — Unnecessary Exposure of Sensitive Data via the User Interface	MASTG-TEST-0276 — Use of the iOS General Pasteboard	Exception Identified	IOS-012	EV-IOS-016
MASVS-PLATFORM-3	MASWE-0038 — Insufficient Protection of Sensitive Data from Screenshots or Screen Recordings	MASTG-TEST-0290 — Runtime Verification of Sensitive Content Exposure in Screenshots During App Backgrounding	Exception Identified	IOS-015	EV-IOS-017
MASVS-PLATFORM-1	MASWE-0002 — Sensitive Data Stored Unencrypted Outside of Private Storage (closest verified concept — primary concern is uncontrolled file-sharing exposure, not the absence of encryption)	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	Exception Identified	IOS-016	EV-IOS-018
MASVS-PLATFORM-1	MASWE-0001 — Sensitive Data Stored Unencrypted in Private Storage (closest verified concept — primary concern is excessive data scope handed to the extension, not the absence of encryption)	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	Exception Identified	IOS-017	EV-IOS-019
MASVS-RESILIENCE-1	MASWE-0051 — Root/Jailbreak Detection Not Implemented	MASTG-TEST-0240 — Jailbreak Detection in Code	Exception Identified	IOS-018	EV-IOS-020
MASVS-PLATFORM-3	MASWE-0037 — Unnecessary Exposure of Sensitive Data via Notifications	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	Exception Identified	IOS-019	EV-IOS-021
MASVS-AUTH-1	MASWE-0024 — Sensitive Data Accessible After Session Termination (closest verified concept — see note)	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	Exception Identified	IOS-020	EV-IOS-022
MASVS-NETWORK-1	MASWE-0026 — Network Traffic Not Encrypted	MASTG-TEST-0342 — References to Weak ATS TLS Policy Exceptions in Info.plist	Exception Identified	IOS-021	EV-IOS-023
MASVS-PRIVACY-1	MASWE-0066 — Inadequate Permission Management	MASTG-TEST-0360 — Purpose String Accuracy for Reachable Protected Resource Access	Exception Identified	IOS-022	EV-IOS-024
MASVS-PRIVACY-2	MASWE-0068 — Incorrect Use of Identifiers for User Tracking	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	Exception Identified	IOS-023	EV-IOS-025
MASVS-CRYPTO-1	MASWE-0012 — Improper Random Number Generation	MASTG-TEST-0311 — Insecure Random API Usage	Exception Identified	IOS-024	EV-IOS-026
MASVS-CODE-4	MASWE-0061 — Debug Artifacts Not Removed	MASTG-TEST-0358 — Implementation Details Exposure Through Logging APIs	Exception Identified	IOS-025	EV-IOS-027
MASVS-STORAGE-1	MASWE-0001 — Sensitive Data Stored Unencrypted in Private Storage	MASTG-TEST-0300 — References to APIs for Storing Unencrypted Data in Private Storage	Exception Identified	IOS-026	EV-IOS-028
MASVS-NETWORK-1	MASWE-0026 — Network Traffic Not Encrypted (closest verified concept — see note; primary concern is ATS exception domain-level granularity, not a specific validation defect)	MASTG-TEST-0342 — References to Weak ATS TLS Policy Exceptions in Info.plist	Exception Identified	IOS-027	EV-IOS-029
MASVS-PRIVACY-1	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	Exception Identified	IOS-028	EV-IOS-030
MASVS-CODE-1	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	Exception Identified	IOS-029	EV-IOS-031
MASVS-STORAGE-2	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	MASTG-TEST-0297 — Sensitive Data Exposure Through Logging APIs	Exception Identified	IOS-030	EV-IOS-032
MASVS-CODE-2	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	Exception Identified	IOS-031	EV-IOS-033
MASVS-PLATFORM-1	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	Exception Identified	IOS-032	EV-IOS-034
MASVS-CODE-4	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	Exception Identified	IOS-033	EV-IOS-035
MASVS-CODE-1	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	Exception Identified	IOS-034	EV-IOS-036
MASVS-RESILIENCE-4	MASWE-0045 — Compiler-Provided Security Features Not Used	MASTG-TEST-0228 / MASTG-TEST-0229 — Position Independent Code (PIC) and Stack Canaries Not Enabled	Exception Identified	IOS-035	EV-IOS-037
MASVS-AUTH-1	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No Exception Identified	IOS-036	EV-IOS-038
MASVS-NETWORK-1	MASWE-0028 — Insecure Identity Pinning	MASTG-TEST-0385 — Missing Certificate Pinning in ATS	No Exception Identified	IOS-037	EV-IOS-039

MASVS Control	MASWE Weakness	MASTG Test	Result	Finding	Evidence
MASVS-RESILIENCE-4	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No Exception Identified	IOS-038	EV-IOS-040
MASVS-RESILIENCE-2	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No Exception Identified	IOS-039	EV-IOS-041
MASVS-STORAGE-1	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No Exception Identified	IOS-040	EV-IOS-042
MASVS-AUTH-2	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No Exception Identified	IOS-041	EV-IOS-043
MASVS-CODE-2	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No Exception Identified	IOS-042	EV-IOS-044
MASVS-PRIVACY-2	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No Exception Identified	IOS-043	EV-IOS-045
MASVS-CODE-1	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No Exception Identified	IOS-044	EV-IOS-046
MASVS-PLATFORM-1	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No Exception Identified	IOS-045	EV-IOS-047
MASVS-CODE-2	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No Exception Identified	IOS-046	EV-IOS-048
MASVS-CODE-1	No single verified MASWE/MASTG identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No single verified MASTG-TEST identifier maps precisely to this scenario; assessed against the relevant MASVS control concept.	No Exception Identified	IOS-047	EV-IOS-049
MASVS-STORAGE-1	N/A	MASTG-TEST-0298 — Runtime Monitoring of Files Eligible for Backup	No Exception Identified	—	—
MASVS-STORAGE-1	N/A	N/A — no single verified MASTG-TEST identifier maps precisely to this scenario (addressed at the development stage; see MASTG-KNOW-0103)	No Exception Identified	—	—
MASVS-STORAGE-2	N/A	MASTG-TEST-0215 — Sensitive Data Not Marked For Backup Exclusion	No Exception Identified	—	—
MASVS-CRYPTO-1	N/A	N/A — no single verified MASTG-TEST identifier maps precisely to this scenario (decomposed into narrower MASTG v2 tests already reflected elsewhere in this table)	No Exception Identified	—	—
MASVS-CRYPTO-2	N/A	MASTG-TEST-0209 — Insufficient Key Sizes	No Exception Identified	—	—
MASVS-AUTH-2	N/A	MASTG-TEST-0266 — References to APIs for Event-Bound Biometric Authentication	No Exception Identified	—	—
MASVS-AUTH-2	N/A	MASTG-TEST-0268 — References to APIs Allowing Fallback to Non-Biometric Authentication	No Exception Identified	—	—
MASVS-NETWORK-1	N/A	MASTG-TEST-0322 — App Transport Security Configurations Allowing Cleartext Traffic	No Exception Identified	—	—
MASVS-NETWORK-1	N/A	MASTG-TEST-0342 — References to Weak ATS TLS Policy Exceptions in Info.plist	No Exception Identified	—	—
MASVS-NETWORK-2	N/A	MASTG-TEST-0385 — Missing Certificate Pinning in ATS	No Exception Identified	—	—
MASVS-PLATFORM-1	N/A	N/A — no single verified MASTG-TEST identifier maps precisely to this scenario (no MASTG v2 successor by design; see deprecation note)	No Exception Identified	—	—
MASVS-PLATFORM-2	N/A	N/A — no single verified MASTG-TEST identifier maps precisely to this scenario	No Exception Identified	—	—
MASVS-PLATFORM-3	N/A	MASTG-TEST-0277 — Sensitive Data in the iOS General Pasteboard at Runtime	No Exception Identified	—	—
MASVS-PLATFORM-1	N/A	MASTG-TEST-0395 — Missing Input Validation in Universal Link Handlers	No Exception Identified	—	—
MASVS-CODE-3	N/A	N/A — no single verified MASTG-TEST identifier maps precisely to this scenario	No Exception Identified	—	—
MASVS-CODE-1	N/A	MASTG-TEST-0229 — Stack Canaries Not Enabled	No Exception Identified	—	—
MASVS-RESILIENCE-1	N/A	MASTG-TEST-0220 — Usage of Outdated Code Signature Format	No Exception Identified	—	—
MASVS-RESILIENCE-4	N/A	N/A — no single verified MASTG-TEST identifier maps precisely to this scenario	No Exception Identified	—	—
MASVS-RESILIENCE-4	N/A	N/A — no single verified MASTG-TEST identifier maps precisely to this scenario	No Exception Identified	—	—
MASVS-RESILIENCE-3	N/A	N/A — no single verified MASTG-TEST identifier maps precisely to this scenario	No Exception Identified	—	—

89 APPENDIX J — FINDING REGISTER

The complete register of all 47 fictional findings identified during this illustrative assessment is provided below, sorted by severity (Critical to Informational) and then by finding ID, for traceability against the detailed findings sections and the Management Action Plan.

Finding ID	Severity	Title	CVSS	Status	Owner	MASVS
IOS-001	CRITICAL	Sensitive Authentication Token Exposure Through Insecure Local Storage	9.0	Remediation In Progress	Mobile Engineering Lead (Fictional Role)	MASVS-STORAGE-1
IOS-002	CRITICAL	Insecure Keychain Access Control Allows Unauthorized Credential Access	8.7	Remediation In Progress	Mobile Engineering Lead (Fictional Role)	MASVS-STORAGE-1
IOS-003	CRITICAL	Broken Authorization in Mobile API Workflow	9.1	Remediation In Progress	Backend/API Engineering Lead (Fictional Role)	MASVS-AUTH-1
IOS-004	CRITICAL	Insecure Universal Link / Deep-Link Flow Enables Sensitive Account Action	8.6	Remediation In Progress	Mobile Engineering Lead (Fictional Role)	MASVS-PLATFORM-1
IOS-005	HIGH	Insecure Custom URL Scheme Enables Authentication or Account Flow Hijacking	7.8	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-PLATFORM-1
IOS-006	HIGH	WKWebView JavaScript Bridge Exposes Sensitive Native Functionality	7.6	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-PLATFORM-2
IOS-007	HIGH	Weak Certificate Validation / Trust Evaluation Logic	7.4	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-NETWORK-2
IOS-008	HIGH	Sensitive Data Exposure Through Application Logs	7.1	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-STORAGE-2
IOS-009	HIGH	Hardcoded API Secret / Credential Material in iOS Binary	7.3	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-STORAGE-1
IOS-010	HIGH	Insecure App Group Data Sharing	7.0	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-STORAGE-1
IOS-011	HIGH	Keychain Access Group Misconfiguration	6.9	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-PLATFORM-1
IOS-013	HIGH	Biometric Authentication Bypass / Improper LocalAuthentication Enforcement	7.2	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-AUTH-2
IOS-014	HIGH	Missing Refresh Token Revocation After Logout	6.8	Remediation Planned	Backend/API Engineering Lead (Fictional Role)	MASVS-AUTH-1
IOS-012	MEDIUM	Sensitive Data Exposure Through Pasteboard	5.4	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-PLATFORM-3
IOS-015	MEDIUM	Sensitive Data Exposure Through Background App Snapshot	5.2	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-PLATFORM-3
IOS-016	MEDIUM	Insecure Document Interaction / File Sharing Configuration	5.0	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-PLATFORM-1
IOS-017	MEDIUM	Improper App Extension Data Isolation	5.1	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-PLATFORM-1
IOS-018	MEDIUM	Weak Jailbreak / Runtime Integrity Protection	5.3	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-RESILIENCE-1
IOS-019	MEDIUM	Sensitive Push Notification Data Exposure	4.9	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-PLATFORM-3
IOS-020	MEDIUM	Weak Session Timeout Enforcement	4.8	Remediation Planned	Backend/API Engineering Lead (Fictional Role)	MASVS-AUTH-1
IOS-021	MEDIUM	Insecure ATS Exception Configuration Permits Weak TLS on Secondary Endpoint	5.0	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-NETWORK-1
IOS-022	MEDIUM	Excessive Privacy Permission Requests Beyond Feature Requirements	4.6	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-PRIVACY-1
IOS-023	MEDIUM	Analytics SDK Persistent Identifier Reuse Across Reinstall	4.7	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-PRIVACY-2
IOS-024	MEDIUM	Insecure Random Number Generation for a Security-Relevant Value	4.9	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-CRYPTO-1
IOS-025	MEDIUM	Verbose Error Handling Reveals Internal Implementation Details	4.5	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-CODE-4
IOS-026	MEDIUM	Insecure Object Persistence of Sensitive Local Cache	5.0	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-STORAGE-1
IOS-027	MEDIUM	Missing App Transport Security Domain-Level Granularity	4.4	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-NETWORK-1
IOS-028	LOW	Unused and Unnecessary Privacy-Sensitive Permission Declarations	3.1	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-PRIVACY-1
IOS-029	LOW	Use of Deprecated iOS APIs With Superseded Secure Alternatives	3.4	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-CODE-1
IOS-030	LOW	Logging Hygiene Weaknesses in Non-Sensitive Diagnostic Modules	3.2	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-STORAGE-2
IOS-031	LOW	Third-Party Dependency Governance Gaps	3.5	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-CODE-2
IOS-032	LOW	Attack-Surface Hardening Opportunity — Excessive Registered URL Scheme and Extension Count	3.0	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-PLATFORM-1
IOS-033	LOW	Missing Automated Security Regression Test Coverage	2.9	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-CODE-4
IOS-034	LOW	Incomplete Secure-Coding Documentation for iOS Engineering	2.7	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-CODE-1
IOS-035	LOW	Missing CI Verification of Stack-Canary and PIE Binary Protections	3.3	Remediation Planned	Mobile Engineering Lead (Fictional Role)	MASVS-RESILIENCE-4
IOS-036	INFORMATIONAL	Positive Observation — Multi-Factor Authentication Enforced on Account Recovery	N/A	Observation Only	Mobile Engineering Lead (Fictional Role)	MASVS-AUTH-1
IOS-037	INFORMATIONAL	Positive Observation — Strong TLS Configuration on Primary API Endpoints	N/A	Observation Only	Mobile Engineering Lead (Fictional Role)	MASVS-NETWORK-1
IOS-038	INFORMATIONAL	Recommendation — Adopt DeviceCheck or App Attest for Server-Side Device Assurance	N/A	Observation Only	Mobile Engineering Lead (Fictional Role)	MASVS-RESILIENCE-4
IOS-039	INFORMATIONAL	Recommendation — Evaluate Binary Obfuscation for Sensitive Business Logic	N/A	Observation Only	Mobile Engineering Lead (Fictional Role)	MASVS-RESILIENCE-2
IOS-040	INFORMATIONAL	Recommendation — Formalize Secrets Management for Build-Time Configuration	N/A	Observation Only	Mobile Engineering Lead (Fictional Role)	MASVS-STORAGE-1

Finding ID	Severity	Title	CVSS	Status	Owner	MASVS
IOS-041	INFORMATIONAL	Recommendation — Add Step-Up Biometric Re-Authentication for High-Value Transactions	N/A	Observation Only	Mobile Engineering Lead (Fictional Role)	MASVS-AUTH-2
IOS-042	INFORMATIONAL	Recommendation — Formalize Third-Party SDK Inventory Tracking	N/A	Observation Only	Mobile Engineering Lead (Fictional Role)	MASVS-CODE-2
IOS-043	INFORMATIONAL	Recommendation — Review Crash-Reporting Payloads for Incidental PII	N/A	Observation Only	Mobile Engineering Lead (Fictional Role)	MASVS-PRIVACY-2
IOS-044	INFORMATIONAL	Recommendation — Establish a Secure-SDLC Checkpoint for Mobile Releases	N/A	Observation Only	Mobile Engineering Lead (Fictional Role)	MASVS-CODE-1
IOS-045	INFORMATIONAL	Observation — Siri Shortcuts and App Intents Surface Is Appropriately Restricted	N/A	Observation Only	Mobile Engineering Lead (Fictional Role)	MASVS-PLATFORM-1
IOS-046	INFORMATIONAL	Recommendation — Increase Dependency-Scanning Cadence for Third-Party SDKs	N/A	Observation Only	Mobile Engineering Lead (Fictional Role)	MASVS-CODE-2
IOS-047	INFORMATIONAL	Recommendation — Document Provisioning Profile and Signing Certificate Lifecycle Management	N/A	Observation Only	Mobile Engineering Lead (Fictional Role)	MASVS-CODE-1

90 APPENDIX K — EVIDENCE REGISTER

The register below documents all 49 fictional evidence references cited throughout the detailed findings, mapped to the finding each item supports. All evidence is synthetic and constructed for this illustrative assessment.

Evidence ID	Type	Description	Finding
EV-IOS-001	Local Storage Extraction	fictional Preferences plist extraction, MASTG-TEST-0300 static/dynamic review notes — item 1 of 2 (Sensitive Authentication Token Exposure Through Insecure Local Storage).	IOS-001
EV-IOS-002	Local Storage Extraction	fictional Preferences plist extraction, MASTG-TEST-0300 static/dynamic review notes — item 2 of 2 (Sensitive Authentication Token Exposure Through Insecure Local Storage).	IOS-001
EV-IOS-003	Keychain Dump	fictional Keychain dump (Insecure Keychain Access Control Allows Unauthorized Credential Access).	IOS-002
EV-IOS-004	Local Backup Extraction	fictional backup extraction listing (Insecure Keychain Access Control Allows Unauthorized Credential Access).	IOS-002
EV-IOS-005	Proxy Capture	fictional proxy capture, transfer-confirmation BOLA reproduction (Broken Authorization in Mobile API Workflow).	IOS-003
EV-IOS-006	Universal Link Trigger / Console Capture	fictional Universal Link trigger and console capture (Insecure Universal Link / Deep-Link Flow Enables Sensitive Account Action).	IOS-004
EV-IOS-007	Custom URL Scheme Collision Test	fictional URL scheme collision test (Insecure Custom URL Scheme Enables Authentication or Account Flow Hijacking).	IOS-005
EV-IOS-008	WKWebView Bridge Capture	fictional bridge exfiltration test (WKWebView JavaScript Bridge Exposes Sensitive Native Functionality).	IOS-006
EV-IOS-009	Proxy Capture	fictional interception proxy capture against statement-export client (Weak Certificate Validation / Trust Evaluation Logic).	IOS-007
EV-IOS-010	Unified Log Capture	fictional unified-log capture during login (Sensitive Data Exposure Through Application Logs).	IOS-008
EV-IOS-011	otool / strings Output	fictional strings extraction of Mach-O binary (Hardcoded API Secret / Credential Material in iOS Binary).	IOS-009
EV-IOS-012	App Group Container Extraction	fictional App Group container extraction (Insecure App Group Data Sharing).	IOS-010
EV-IOS-013	Keychain Dump	fictional Keychain access-group scope test (Keychain Access Group Misconfiguration).	IOS-011
EV-IOS-014	Runtime Instrumentation Test Log	fictional instrumentation bypass of LAContext completion handler (Biometric Authentication Bypass / Improper LocalAuthentication Enforcement).	IOS-013
EV-IOS-015	Token Replay Test Log	fictional post-logout token-replay test (Missing Refresh Token Revocation After Logout).	IOS-014
EV-IOS-016	Pasteboard Polling Capture	fictional pasteboard polling test (Sensitive Data Exposure Through Pasteboard).	IOS-012
EV-IOS-017	App Switcher Snapshot Capture	fictional App Switcher snapshot capture (Sensitive Data Exposure Through Background App Snapshot).	IOS-015
EV-IOS-018	Document Interaction / File-Sharing Browse	fictional Files app / Finder file-sharing browse (Insecure Document Interaction / File Sharing Configuration).	IOS-016
EV-IOS-019	App Extension Static Review	fictional static review of share-extension data access (Improper App Extension Data Isolation).	IOS-017
EV-IOS-020	Jailbreak-Detection Bypass Test Log	fictional jailbreak-detection bypass test (Weak Jailbreak / Runtime Integrity Protection).	IOS-018
EV-IOS-021	Notification Capture	fictional lock-screen notification capture (Sensitive Push Notification Data Exposure).	IOS-019
EV-IOS-022	Session Timeout Test Log	fictional idle-timeout interval test (Weak Session Timeout Enforcement).	IOS-020
EV-IOS-023	Info.plist / ATS Configuration Review	fictional Info.plist ATS exception review (Insecure ATS Exception Configuration Permits Weak TLS on Secondary Endpoint).	IOS-021
EV-IOS-024	Info.plist Permission Review	fictional Info.plist permission review (Excessive Privacy Permission Requests Beyond Feature Requirements).	IOS-022
EV-IOS-025	Identifier Persistence Test Log	fictional delete/reinstall identifier persistence test (Analytics SDK Persistent Identifier Reuse Across Reinstall).	IOS-023
EV-IOS-026	Static Code Review — Cryptography	fictional static review of nonce generation code (Insecure Random Number Generation for a Security-Relevant Value).	IOS-024
EV-IOS-027	API Error Response Capture	fictional malformed-request QA-build error capture (Verbose Error Handling Reveals Internal Implementation Details).	IOS-025
EV-IOS-028	Local Cache Extraction	fictional transaction-cache extraction (Insecure Object Persistence of Sensitive Local Cache).	IOS-026
EV-IOS-029	Info.plist / ATS Configuration Review	fictional ATS configuration review (Missing App Transport Security Domain-Level Granularity).	IOS-027
EV-IOS-030	Info.plist Permission Review	fictional Info.plist and UI-flow cross-reference (Unused and Unnecessary Privacy-Sensitive Permission Declarations).	IOS-028
EV-IOS-031	Symbol Table / Binary Review	fictional symbol table review (Use of Deprecated iOS APIs With Superseded Secure Alternatives).	IOS-029
EV-IOS-032	Test Log / Analysis Note	fictional console log review (Logging Hygiene Weaknesses in Non-Sensitive Diagnostic Modules).	IOS-030
EV-IOS-033	Dependency Manifest / SDK Inventory Review	fictional dependency manifest and SDK inventory review (Third-Party Dependency Governance Gaps).	IOS-031
EV-IOS-034	Info.plist / Extension Bundle Enumeration	fictional Info.plist and extension bundle enumeration (Attack-Surface Hardening Opportunity — Excessive Registered URL Scheme and Extension Count).	IOS-032
EV-IOS-035	CI/CD Configuration Review	fictional CI/CD pipeline configuration review (Missing Automated Security Regression Test Coverage).	IOS-033
EV-IOS-036	Interview Notes	fictional documentation review and interview notes (Incomplete Secure-Coding Documentation for iOS Engineering).	IOS-034
EV-IOS-037	otool / strings Output	fictional otool binary protection review (Missing CI Verification of Stack-Canary and PIE Binary Protections).	IOS-035
EV-IOS-038	Manual Flow Walkthrough	fictional recovery flow walkthrough (Positive Observation — Multi-Factor Authentication Enforced on Account Recovery).	IOS-036
EV-IOS-039	TLS Negotiation Capture	fictional TLS negotiation capture (Positive Observation — Strong TLS Configuration on Primary API Endpoints).	IOS-037
EV-IOS-040	Entitlements / Framework Review	fictional framework/entitlement review (Recommendation — Adopt DeviceCheck or App Attest for Server-Side Device Assurance).	IOS-038
EV-IOS-041	Symbol Table / Binary Review	fictional symbol table review (Recommendation — Evaluate Binary Obfuscation for Sensitive Business Logic).	IOS-039
EV-IOS-042	Build Configuration Review	fictional build configuration review (Recommendation — Formalize Secrets Management for Build-Time Configuration).	IOS-040
EV-IOS-043	Manual Flow Walkthrough	fictional transaction flow walkthrough (Recommendation — Add Step-Up Biometric Re-Authentication for High-Value Transactions).	IOS-041
EV-IOS-044	Interview Notes	fictional interview notes (Recommendation — Formalize Third-Party SDK Inventory Tracking).	IOS-042
EV-IOS-045	SDK Configuration Review	fictional crash-reporting SDK configuration review (Recommendation — Review Crash-Reporting Payloads for Incidental PII).	IOS-043
EV-IOS-046	Interview Notes	fictional interview notes (Recommendation — Establish a Secure-SDLC Checkpoint for Mobile Releases).	IOS-044
EV-IOS-047	App Intents Enumeration	fictional App Intents enumeration (Observation — Siri Shortcuts and App Intents Surface Is Appropriately Restricted).	IOS-045
EV-IOS-048	Interview Notes	fictional interview notes (Recommendation — Increase Dependency-Scanning Cadence for Third-Party SDKs).	IOS-046
EV-IOS-049	codesign / Provisioning Profile Review	fictional codesign and provisioning profile review (Recommendation — Document Provisioning Profile and Signing Certificate Lifecycle Management).	IOS-047

All evidence identifiers and descriptions above are fictional and synthetic — ILLUSTRATIVE / FICTIONAL SAMPLE EVIDENCE.

91 APPENDIX L — TEST ACCOUNTS & DEVICES

The fictional test accounts and devices below were used throughout this assessment. No real passwords are recorded in this report; all credential values are placeholders.

Test Accounts

Fictional Account	Role	Account ID	Credential	Purpose
ios.user01@example.test	Standard User (Test Account A)	ACCT-559002	*[Fictional Test Credential]*	Primary standard-user account — core flow testing and IOS-003 source-account role.
ios.user02@example.test	Standard User (Test Account B)	ACCT-771123	*[Fictional Test Credential]*	Secondary account — cross-account / object-level-authorization isolation testing (IOS-003).
ios.newuser01@example.test	Standard User (New Enrollment)	ACCT-559099	*[Fictional Test Credential]*	Onboarding, MFA enrollment, and biometric-registration flow testing.
ios.qa-admin01@example.test	QA / Internal Test Account	ACCT-000411	*[Fictional Test Credential]*	Used for QA-build verbose-logging capture (IOS-008) and pre-production configuration review.
ios.attacker-sim01@example.test	Standard User (Simulated Low-Privilege)	ACCT-990001	*[Fictional Test Credential]*	Simulated-attacker account for authorization, Universal Link, and custom URL scheme testing.

Test Devices

Fictional Device	Profile	Purpose
Fictional Test Device A	Jailbroken iPhone 13 mini, iOS 16.4 (fictional checkra1n-style jailbreak)	Keychain dump, filesystem extraction, App Group container review, and jailbreak-detection bypass testing.
Fictional Test Device B	Non-Jailbroken iPhone 15 Pro, iOS 18.1	Baseline UX and control-validation testing under standard conditions.
Fictional Test Device C	Non-Jailbroken iPhone SE (3rd generation), iOS 16.0 (minimum-OS boundary)	Minimum-deployment-target behavior and backward-compatibility testing.
Fictional Test Device D	Jailbroken iPhone 12, iOS 17.5 (fictional jailbreak-hiding tweak installed)	Runtime instrumentation (Frida-style hooking), biometric-bypass, and jailbreak-hiding-tweak evasion testing.
Fictional Proxy Host	Interception Proxy (Burp-style) on Isolated Test Network	Network traffic interception, ATS/TLS, and certificate-pinning-resistance testing.
Fictional macOS Test Host	Paired macOS workstation (Xcode, Console.app, log CLI, libimobiledevice-equivalent tooling)	IPA extraction, otool/strings binary analysis, unified-log capture, and unencrypted local backup extraction.

All accounts, devices, identifiers, and credential placeholders above are fictional. No real credentials are recorded anywhere in this report.

92 APPENDIX M — THIRD-PARTY SDK INVENTORY

The register below documents all 10 fictional third-party SDKs and frameworks embedded in NovaSpherePay-6.4.2.ipa, with vendor, purpose, data access, governance notes, and any corresponding finding.

SDK / Framework	Vendor (Fictional)	Purpose	Data Access	Governance Notes	Finding
MapSDKKit	Fictional Third-Party Vendor A	Branch/ATM-locator map rendering	Coarse-to-fine device location (feature-scoped)	Core SDK — documented owner and review cadence.	IOS-009
AnalyticsKit	Fictional Third-Party Vendor B	Usage analytics, funnel tracking, feature-adoption metrics	Persistent tracking identifier, device metadata, usage events	Core SDK — documented owner; identifier-reset gap tracked separately.	IOS-023, IOS-042
CrashReportKit	Fictional Third-Party Vendor C	Crash and non-fatal error reporting	Stack traces, breadcrumb events, device metadata	Core SDK — documented owner; breadcrumb PII review recommended.	IOS-043
PartnerLinkKit	Fictional Third-Party Vendor E	Bank-account aggregation OAuth account-linking flow	OAuth authorization code, linked-account metadata	Core SDK — documented owner; callback-transport hardening recommended.	IOS-005
DocExportKit (legacy)	Fictional Third-Party Vendor G	Legacy statement PDF export backend integration	Fictional account statement data (PDF payload)	Legacy integration — weak trust evaluation and reduced TLS floor tracked separately.	IOS-007, IOS-021
ImageCacheKit	Fictional Third-Party Vendor D	Remote avatar/promo image loading and caching	Image URLs only; no account data	Lower-usage SDK — no documented owner or review cadence.	IOS-031
Legacy Analytics Helper	Fictional Third-Party Vendor H (deprecated)	Supplementary funnel analytics predating AnalyticsKit's adoption	Usage events, device metadata	Lower-usage/deprecated SDK — no documented owner or review cadence.	IOS-031
Biometric UX Wrapper	Fictional Third-Party Vendor I	Thin convenience wrapper around Apple LocalAuthentication for consistent biometric-prompt UX	None (local authentication only; no data leaves the device)	Core SDK — documented owner.	IOS-013
UI Component Kit	Fictional Third-Party Vendor J	Shared design-system UI components (buttons, form fields, charts)	None	Core SDK — documented owner; no data-access concerns identified.	—
Push Delivery Helper	Fictional Third-Party Vendor F	Thin wrapper simplifying APNs registration and payload handling	Push token, notification payload metadata	Core SDK — documented owner.	IOS-019

93 APPENDIX N — REMEDIATION PRIORITY MATRIX

The matrix below consolidates all 47 fictional findings into a single remediation-priority view, derived directly from the finding register, with remediation timeline, accountable owner, target date, and current status for each finding.

Finding ID	Severity	Title	Remediation Priority	Owner	Target Date	Status
IOS-001	CRITICAL	Sensitive Authentication Token Exposure Through Insecure Local Storage	Immediate — 0-30 Days	Mobile Engineering Lead (Fictional Role)	10 October 2026	Remediation In Progress
IOS-002	CRITICAL	Insecure Keychain Access Control Allows Unauthorized Credential Access	Immediate — 0-30 Days	Mobile Engineering Lead (Fictional Role)	10 October 2026	Remediation In Progress
IOS-003	CRITICAL	Broken Authorization in Mobile API Workflow	Immediate — 0-30 Days	Backend/API Engineering Lead (Fictional Role)	24 September 2026	Remediation In Progress
IOS-004	CRITICAL	Insecure Universal Link / Deep-Link Flow Enables Sensitive Account Action	Immediate — 0-30 Days	Mobile Engineering Lead (Fictional Role)	10 October 2026	Remediation In Progress
IOS-005	HIGH	Insecure Custom URL Scheme Enables Authentication or Account Flow Hijacking	31-60 Days	Mobile Engineering Lead (Fictional Role)	15 November 2026	Remediation Planned
IOS-006	HIGH	WKWebView JavaScript Bridge Exposes Sensitive Native Functionality	31-60 Days	Mobile Engineering Lead (Fictional Role)	15 November 2026	Remediation Planned
IOS-007	HIGH	Weak Certificate Validation / Trust Evaluation Logic	31-60 Days	Mobile Engineering Lead (Fictional Role)	15 November 2026	Remediation Planned
IOS-008	HIGH	Sensitive Data Exposure Through Application Logs	31-60 Days	Mobile Engineering Lead (Fictional Role)	15 November 2026	Remediation Planned
IOS-009	HIGH	Hardcoded API Secret / Credential Material in iOS Binary	31-60 Days	Mobile Engineering Lead (Fictional Role)	15 November 2026	Remediation Planned
IOS-010	HIGH	Insecure App Group Data Sharing	31-60 Days	Mobile Engineering Lead (Fictional Role)	15 November 2026	Remediation Planned
IOS-011	HIGH	Keychain Access Group Misconfiguration	31-60 Days	Mobile Engineering Lead (Fictional Role)	15 November 2026	Remediation Planned
IOS-013	HIGH	Biometric Authentication Bypass / Improper LocalAuthentication Enforcement	31-60 Days	Mobile Engineering Lead (Fictional Role)	15 November 2026	Remediation Planned
IOS-014	HIGH	Missing Refresh Token Revocation After Logout	31-60 Days	Backend/API Engineering Lead (Fictional Role)	15 November 2026	Remediation Planned
IOS-012	MEDIUM	Sensitive Data Exposure Through Pasteboard	61-90 Days	Mobile Engineering Lead (Fictional Role)	15 December 2026	Remediation Planned
IOS-015	MEDIUM	Sensitive Data Exposure Through Background App Snapshot	61-90 Days	Mobile Engineering Lead (Fictional Role)	15 December 2026	Remediation Planned
IOS-016	MEDIUM	Insecure Document Interaction / File Sharing Configuration	61-90 Days	Mobile Engineering Lead (Fictional Role)	15 December 2026	Remediation Planned
IOS-017	MEDIUM	Improper App Extension Data Isolation	61-90 Days	Mobile Engineering Lead (Fictional Role)	15 December 2026	Remediation Planned
IOS-018	MEDIUM	Weak Jailbreak / Runtime Integrity Protection	61-90 Days	Mobile Engineering Lead (Fictional Role)	15 December 2026	Remediation Planned
IOS-019	MEDIUM	Sensitive Push Notification Data Exposure	61-90 Days	Mobile Engineering Lead (Fictional Role)	15 December 2026	Remediation Planned
IOS-020	MEDIUM	Weak Session Timeout Enforcement	61-90 Days	Backend/API Engineering Lead (Fictional Role)	15 December 2026	Remediation Planned
IOS-021	MEDIUM	Insecure ATS Exception Configuration Permits Weak TLS on Secondary Endpoint	61-90 Days	Mobile Engineering Lead (Fictional Role)	15 December 2026	Remediation Planned
IOS-022	MEDIUM	Excessive Privacy Permission Requests Beyond Feature Requirements	61-90 Days	Mobile Engineering Lead (Fictional Role)	15 December 2026	Remediation Planned
IOS-023	MEDIUM	Analytics SDK Persistent Identifier Reuse Across Reinstall	61-90 Days	Mobile Engineering Lead (Fictional Role)	15 December 2026	Remediation Planned
IOS-024	MEDIUM	Insecure Random Number Generation for a Security-Relevant Value	61-90 Days	Mobile Engineering Lead (Fictional Role)	15 December 2026	Remediation Planned
IOS-025	MEDIUM	Verbose Error Handling Reveals Internal Implementation Details	61-90 Days	Mobile Engineering Lead (Fictional Role)	15 December 2026	Remediation Planned
IOS-026	MEDIUM	Insecure Object Persistence of Sensitive Local Cache	61-90 Days	Mobile Engineering Lead (Fictional Role)	15 December 2026	Remediation Planned
IOS-027	MEDIUM	Missing App Transport Security Domain-Level Granularity	61-90 Days	Mobile Engineering Lead (Fictional Role)	15 December 2026	Remediation Planned
IOS-028	LOW	Unused and Unnecessary Privacy-Sensitive Permission Declarations	90+ Days	Mobile Engineering Lead (Fictional Role)	31 January 2027	Remediation Planned
IOS-029	LOW	Use of Deprecated iOS APIs With Superseded Secure Alternatives	90+ Days	Mobile Engineering Lead (Fictional Role)	31 January 2027	Remediation Planned
IOS-030	LOW	Logging Hygiene Weaknesses in Non-Sensitive Diagnostic Modules	90+ Days	Mobile Engineering Lead (Fictional Role)	31 January 2027	Remediation Planned
IOS-031	LOW	Third-Party Dependency Governance Gaps	90+ Days	Mobile Engineering Lead (Fictional Role)	31 January 2027	Remediation Planned
IOS-032	LOW	Attack-Surface Hardening Opportunity — Excessive Registered URL Scheme and Extension Count	90+ Days	Mobile Engineering Lead (Fictional Role)	31 January 2027	Remediation Planned
IOS-033	LOW	Missing Automated Security Regression Test Coverage	90+ Days	Mobile Engineering Lead (Fictional Role)	31 January 2027	Remediation Planned
IOS-034	LOW	Incomplete Secure-Coding Documentation for iOS Engineering	90+ Days	Mobile Engineering Lead (Fictional Role)	31 January 2027	Remediation Planned
IOS-035	LOW	Missing CI Verification of Stack-Canary and PIE Binary Protections	90+ Days	Mobile Engineering Lead (Fictional Role)	31 January 2027	Remediation Planned
IOS-036	INFORMATIONAL	Positive Observation — Multi-Factor Authentication Enforced on Account Recovery	N/A — Informational	Mobile Engineering Lead (Fictional Role)	N/A	Observation Only
IOS-037	INFORMATIONAL	Positive Observation — Strong TLS Configuration on Primary API Endpoints	N/A — Informational	Mobile Engineering Lead (Fictional Role)	N/A	Observation Only
IOS-038	INFORMATIONAL	Recommendation — Adopt DeviceCheck or App Attest for Server-Side Device Assurance	N/A — Informational	Mobile Engineering Lead (Fictional Role)	N/A	Observation Only
IOS-039	INFORMATIONAL	Recommendation — Evaluate Binary Obfuscation for Sensitive Business Logic	N/A — Informational	Mobile Engineering Lead (Fictional Role)	N/A	Observation Only
IOS-040	INFORMATIONAL	Recommendation — Formalize Secrets Management for Build-Time Configuration	N/A — Informational	Mobile Engineering Lead (Fictional Role)	N/A	Observation Only
IOS-041	INFORMATIONAL	Recommendation — Add Step-Up Biometric Re-Authentication for High-Value Transactions	N/A — Informational	Mobile Engineering Lead (Fictional Role)	N/A	Observation Only

Finding ID	Severity	Title	Remediation Priority	Owner	Target Date	Status
IOS-042	INFORMATIONAL	Recommendation — Formalize Third-Party SDK Inventory Tracking	N/A — Informational	Mobile Engineering Lead (Fictional Role)	N/A	Observation Only
IOS-043	INFORMATIONAL	Recommendation — Review Crash-Reporting Payloads for Incidental PII	N/A — Informational	Mobile Engineering Lead (Fictional Role)	N/A	Observation Only
IOS-044	INFORMATIONAL	Recommendation — Establish a Secure-SDLC Checkpoint for Mobile Releases	N/A — Informational	Mobile Engineering Lead (Fictional Role)	N/A	Observation Only
IOS-045	INFORMATIONAL	Observation — Siri Shortcuts and App Intents Surface Is Appropriately Restricted	N/A — Informational	Mobile Engineering Lead (Fictional Role)	N/A	Observation Only
IOS-046	INFORMATIONAL	Recommendation — Increase Dependency-Scanning Cadence for Third-Party SDKs	N/A — Informational	Mobile Engineering Lead (Fictional Role)	N/A	Observation Only
IOS-047	INFORMATIONAL	Recommendation — Document Provisioning Profile and Signing Certificate Lifecycle Management	N/A — Informational	Mobile Engineering Lead (Fictional Role)	N/A	Observation Only

SEO / AEO LANDING PAGE CONTENT BLUEPRINT

Internal marketing/content reference — not part of the technical assessment itself

This appendix-style section provides TMG Security's marketing and content team with a ready-to-use blueprint for publishing this sample report as a landing page, optimized for both traditional search (SEO) and AI-answer-engine extraction (AEO). It is written for internal reference only and does not describe any additional technical testing.

Page Metadata

Field	Recommended Value
H1 (On-Page Title)	iOS Application Penetration Testing Report — Sample
SEO Title (Title Tag)	iOS Penetration Testing Report Sample | TMG Security
Meta Description	See a full example iOS penetration testing report: 47 illustrative findings, OWASP MASVS/MASTG/MASWE mapping, evidence, and remediation guidance from TMG Security.
Target URL	/security-assessment-reports/ios-application-pentesting-sample-report/
Primary Keyword	iOS penetration testing report sample

Secondary Keywords

- iOS app security assessment
- iOS MASVS pentest report example
- mobile app penetration test sample PDF
- OWASP MASTG iOS testing report
- iOS application security audit example
- Keychain security testing report
- mobile application penetration testing sample report
- iOS app pentest findings example

Answer-First Intro (for AI answer engines)

This sample iOS application penetration testing report shows what a professional TMG Security mobile assessment deliverable looks like: a 47-finding illustrative report covering Keychain and local storage security, authentication and biometric controls, network and certificate validation, custom URL scheme and Universal Link handling, App Extensions and App Groups, and full OWASP MASVS, MASTG, and MASWE traceability. It is a fictional, illustrative example — not a real client engagement — intended to help prospective clients understand TMG Security's iOS testing methodology and reporting depth before commissioning an assessment.

This is a fictional sample report. It does not describe a real client, application, or vulnerability, and TMG Security is not affiliated with, endorsed by, or certified by OWASP.

Recommended Page Outline (H2 / H3 Structure)

- H2: What This Sample Report Covers
- H3: Scope & Methodology
- H3: OWASP MASVS / MASTG / MASWE Coverage
- H2: Example Findings by Severity
- H3: Critical & High Findings
- H3: Medium, Low & Informational Findings
- H2: Appendices Included in the Full Report
- H2: How TMG Security's iOS Testing Methodology Works
- H2: Frequently Asked Questions
- H2: Related Services
- H2: Request a Real Assessment (CTA)

Frequently Asked Questions (FAQPage Schema Candidates)

What does an iOS penetration testing report include?

A professional iOS penetration testing report typically includes an executive summary, methodology, OWASP MASVS/MASTG/MASWE coverage mapping, detailed findings with evidence and CVSS-style scoring, a remediation roadmap, and supporting appendices such as test case and API endpoint registers. This sample report illustrates each of those components using fictional data.

What is OWASP MASVS and MASTG for iOS?

The OWASP Mobile Application Security Verification Standard (MASVS) defines security requirements for mobile apps, while the Mobile Application Security Testing Guide (MASTG) provides corresponding test cases; MASWE catalogs common mobile app weaknesses. TMG Security maps its iOS findings to these frameworks for traceability, without claiming any OWASP affiliation, endorsement, or certification.

Is this a real client report?

No. This is a fictional, illustrative sample report built around a fictional company and application to demonstrate TMG Security's reporting depth and methodology. No real client, vulnerability, or credential is described anywhere in this document.

How long does an iOS penetration test take?

Timelines vary with app complexity, API surface size, and scope, but a typical iOS mobile application assessment (static, dynamic, and API testing) runs several weeks from kickoff to draft report. Speak with TMG Security for a scoped estimate.

What is included in Keychain and local storage testing?

Keychain and local storage testing reviews how sensitive data — authentication tokens, biometric-vault items, cached records — is stored on-device, including Keychain accessibility classes, access-control policies, Keychain access groups, and any non-Keychain local storage such as property lists or unencrypted caches.

Does TMG Security test Universal Links and custom URL schemes?

Yes. iOS mobile assessments cover both custom URL scheme and Universal Link handling, checking for insufficient source verification, missing in-app confirmation on sensitive actions, and scheme-hijacking risk from co-installed applications.

What is App Intents / AI agent exposure testing?

As Siri Shortcuts and App Intents increasingly expose app functionality to voice assistants and AI agents, testing reviews which actions are reachable through this surface and whether sensitive, state-changing actions (such as funds transfers) are appropriately excluded from voice/agent invocation.

How is severity determined in this report?

Each finding receives an illustrative CVSS-inspired score and a qualitative severity rating (Critical, High, Medium, Low, or Informational) based on exploitability, business impact, and data sensitivity, consistent with the methodology described in the Risk Rating Methodology section of the full report.

Can I request a real iOS penetration test from TMG Security?

Yes. This sample report is intended to help prospective clients understand TMG Security's iOS testing approach before commissioning a real, scoped engagement. Use the contact details in the report's closing section to start that conversation.

Internal Linking Recommendations

- /services/mobile-application-security/ — primary service page for mobile app pentesting
- /services/api-security/ — for the mobile API authorization and integration findings
- /services/web-application-security/ — cross-link for WKWebView-related content
- /services/penetration-testing/ — general penetration testing service hub
- /services/red-team/ — for readers interested in broader adversarial testing
- /services/ai-llm-security/ — cross-link from the App Intents / AI agent exposure discussion
- /security-assessment-reports/ — hub page linking to all TMG Security sample reports, including the companion Android sample report

Call-to-Action Copy

"See what a real iOS penetration test could uncover in your app. Talk to TMG Security about scoping a full assessment."

Image Alt Text Suggestions

- Alt: "Executive risk dashboard from a sample iOS penetration testing report showing 47 illustrative findings by severity"
- Alt: "Illustrative iOS application architecture diagram used in a TMG Security sample penetration test report"
- Alt: "Sample MASVS coverage chart showing OWASP Mobile Application Security Verification Standard categories tested"
- Alt: "Illustrative risk heatmap from a sample iOS penetration testing report"

Breadcrumb Structure

Home > Security Assessment Reports > iOS Application Penetration Testing Sample Report

Schema.org Recommendations

Use only the following schema types for this page: WebPage, BreadcrumbList, Organization, WebSite, and FAQPage (populated from the FAQ entries above). Do NOT use Review, Rating, Product, Certification, or SecurityAudit schema — this page describes a fictional illustrative sample, not a certified audit, a rated product, or a real completed engagement, and using those schema types would misrepresent it.

This blueprint is for internal marketing/content use only and is not part of the technical penetration testing methodology described elsewhere in this report.