CETONIX

SECURITY ASSURANCE · OFFENSIVE SECURITY PRACTICE

CONFIDENTIAL · TLP:AMBER+STRICT

MOBILE APPLICATION PENETRATION TEST

iOS Application · OWASP MASVS / MASTG

Static, Runtime, IPC and Backend Testing

PREPARED FOR

ACME Corporation, Inc.

ACME Mobile for iOS — com.acme.workspace

ENGAGEMENT ID	REPORT VERSION	TEST WINDOW	ISSUED
CTX-PT-2026-0437	1.2 — Final (Post-Retest)	17 - 28 Aug 2026	04 September 2026

REDACTED DISTRIBUTION COPY

Request/response bodies, credentials, tokens, host identifiers and customer data are masked. The client copy contains all evidence in full. See §2 Redaction Key.

Cetonix · EON Freezone, Kharadi, Pune 411014, India · info@cetonix.com

Contents

This report is issued as a redacted distribution copy. Section 2 explains exactly what has been masked and what the client copy contains in its place.

1. Document Control

1.1 Engagement details

Engagement reference	CTX-PT-2026-0437
Client	ACME Corporation, Inc.
Engagement type	iOS application penetration test — static, runtime, IPC and backend testing
Target	ACME Mobile for iOS — com.acme.workspace (TestFlight build)
Testing window	17 August 2026 – 28 August 2026 (10 working days)
Retest window	1 September 2026 – 2 September 2026
Report issued	4 September 2026
Report version	1.2 — Final (post-retest)
Classification	CONFIDENTIAL · TLP:AMBER+STRICT
Testing standard	OWASP MASVS v2.1 · OWASP MASTG · OWASP MASWE · PTES · NIST SP 800-115

1.2 Version history

VER.	DATE	AUTHOR	CHANGE
0.1	26 Aug 2026	Lead Consultant	Initial draft issued for internal peer review
0.9	29 Aug 2026	Technical QA Reviewer	Peer review complete; CVSS vectors validated; two findings re-scored
1.0	31 Aug 2026	Practice Manager	Final draft issued to client for remediation
1.1	2 Sep 2026	Lead Consultant	Retest results incorporated; nine findings verified as resolved
1.2	**4 September 2026**	**Practice Manager**	**Final issue — attestation letter appended; redacted distribution copy produced**

1.3 Distribution and authorised recipients

This document is released to the named recipients below. Onward distribution requires written approval from the client's engagement owner. Cetonix retains no copy of client credentials or captured data beyond the retention period stated in section 2.4.

NAME / ROLE	ORGANISATION	COPY ISSUED
[Client Engagement Owner] Head of Security	ACME Corporation, Inc.	**Client copy — unredacted**
[Client Technical Lead] Head of Mobile Engineering	ACME Corporation, Inc.	**Client copy — unredacted**
[Client Compliance Lead]	ACME Corporation, Inc.	**Client copy — unredacted**

NAME / ROLE	ORGANISATION	COPY ISSUED
Director, GRC		
External auditor / assessor	[Audit firm]	**Redacted distribution copy**
Customer vendor-risk reviewers	Various	**Redacted distribution copy**
Lead Consultant, Test Team	Cetonix	Master copy — encrypted at rest

1.4 Assessment team

ROLE	RESPONSIBILITY	QUALIFICATION PROFILE
Lead Consultant	Test execution, findings, report authorship	Mobile offensive security specialist; iOS reverse engineering and runtime instrumentation; 7 years' practice
Senior Analyst	Static analysis, IPC and backend testing	iOS platform background; MASVS assessment specialisation
Technical QA Reviewer	Independent peer review, CVSS validation	Did not participate in testing; reviews all findings prior to issue
Practice Manager	Engagement governance, client communication	Accountable for scope, impartiality screening and final sign-off

Named team, not a rotating pool

The same lead consultant runs the engagement from scoping through retest and is available to your engineering team throughout. Analyst identities are provided to the client in the unredacted copy and withheld here.

2. Confidentiality, Handling and Redaction Key

This copy has been prepared for onward distribution to parties outside the client's security team — auditors, assessors, prospective customers and vendor-risk reviewers. It carries the full narrative, methodology, findings, severity ratings and remediation guidance of the original, with sensitive technical detail masked so that the document cannot itself be used to attack the platform it describes.

What redaction does and does not remove

Nothing about the existence, severity, impact or remediation of a finding is withheld. What is removed is the material that would let a reader reproduce the attack: live identifiers, credentials, tokens, payload strings, internal hostnames and customer data.

2.1 Redaction key

Every masked value is replaced by a solid bar and a bracketed tag naming what was removed. The tags used in this report are listed below.

TAG	WHAT IT MASKS	IN THE CLIENT COPY
[REDACTED — SESSION TOKEN]	Bearer tokens, JWTs, refresh tokens and cookies captured during testing	Full value, truncated to 32 chars
[REDACTED — CREDENTIAL]	Account passwords, API keys, HMAC secrets and MFA codes	Full value
[REDACTED — TENANT UUID]	Workspace and tenant identifiers that map to real customers	Full identifier
[REDACTED — OBJECT ID]	Record identifiers used to demonstrate direct object reference flaws	Full identifier plus the enumeration range tested
[REDACTED — CUSTOMER PII]	Names, email addresses, billing contacts and any personal data observed	Recorded as a data-class count only; raw PII is never retained
[REDACTED — PAYLOAD]	Working exploit strings and injection payloads	Full payload with encoding notes
[REDACTED — KEYCHAIN ITEM]	Keychain service and account identifiers, and the values they hold	Full item attributes and values
[REDACTED — HARDCODED KEY]	API keys and certificates recovered from the application binary	Full value and its location in the binary
[REDACTED — VICTIM ACCOUNT]	Account identifiers belonging to the second test user	Full identifier
[REDACTED — CREDENTIAL MATERIAL]	Any secret material recovered from device storage or traffic	**Masked in every copy — see 2.3**

2.2 Shared copy versus client copy

THIS COPY — REDACTED FOR DISTRIBUTION

```
POST /api/v2/auth/token HTTP/1.1
Host: api.acme-corp.example
{"email":"                              [ACCOUNT]
  "password":"                          [CREDENTIAL]
  "otp":"                               [MFA CODE]

HTTP/1.1 200 OK
{"access_token":"                       [BEARER TOKEN]
  "refresh_token":"                     [REFRESH TOKEN]
  "tenant":"                            [TENANT UUID]
```

Sanitised for auditors, prospects and vendor-risk reviewers.

CLIENT COPY — COMPLETE EVIDENCE

```
POST /api/v2/auth/token HTTP/1.1
Host: api.acme-corp.example
{"email":"test.analyst+al@cetonix...
  "password":"Tr!alAcct-2026-...
  "otp":"4915"

HTTP/1.1 200 OK
{"access_token":"eyJhbGciOiJIUzI1Ni...
  "refresh_token":"rt_7f3a91c4e0b2...
  "tenant":"9c1e4f70-3ab2-4d18-...
```

Issued to named client contacts over encrypted transfer.

Figure 2.1 — The same authentication capture as it appears in this copy (left) and in the client copy (right).

2.3 Material redacted in every copy

Three classes of data are masked even in the client's own copy, because retaining them would create risk that outlasts the engagement:

- Live credential material recovered from the Keychain or from intercepted traffic — session and refresh tokens, API keys and certificates found in the binary. Cetonix records the type, location and scope, and instructs the client to rotate; the value itself is destroyed at capture.
- Customer personal data read from the application's local stores and caches. Records are counted and classified, never copied off the test device.
- Any payment instrument data. Testing is designed so that primary account numbers are never retrieved; where a payment reference appears it is recorded as a token reference only.

2.4 Handling, retention and destruction

Transfer	Encrypted transfer to named recipients only. Reports are not sent as unprotected email attachments.
Storage	Evidence and drafts held encrypted at rest, accessible only to the assigned engagement team and the QA reviewer.
Retention	Evidence retained for 90 days after the retest closes, to support re-issue and auditor queries; the report itself is retained for the period agreed in the master services agreement.
Destruction	All captures, credentials and test accounts are destroyed on client request or at the end of the retention period, with written confirmation.
Third parties	No client data, capture or credential is shared with, or processed by, any third-party service. See section 10 for the position on AI tooling.

3. Executive Summary

PURPOSE

ACME Corporation engaged Cetonix to perform an independent penetration test of the ACME Mobile application for iOS. The engagement was commissioned alongside the Android assessment so that both mobile clients and their shared backend were evaluated against the same threat model within a single quarter.

OVERALL ASSESSMENT

> **Overall risk rating: MEDIUM-HIGH — reduced to LOW following remediation and retest**
> The iOS build is materially stronger than its Android counterpart: the platform's own protections do a great deal of work, and no finding reached Critical. The weaknesses that remain all follow from using those protections as user-experience features rather than as security boundaries. Nine of thirteen findings, including all three rated High, were remediated and verified during the retest window.

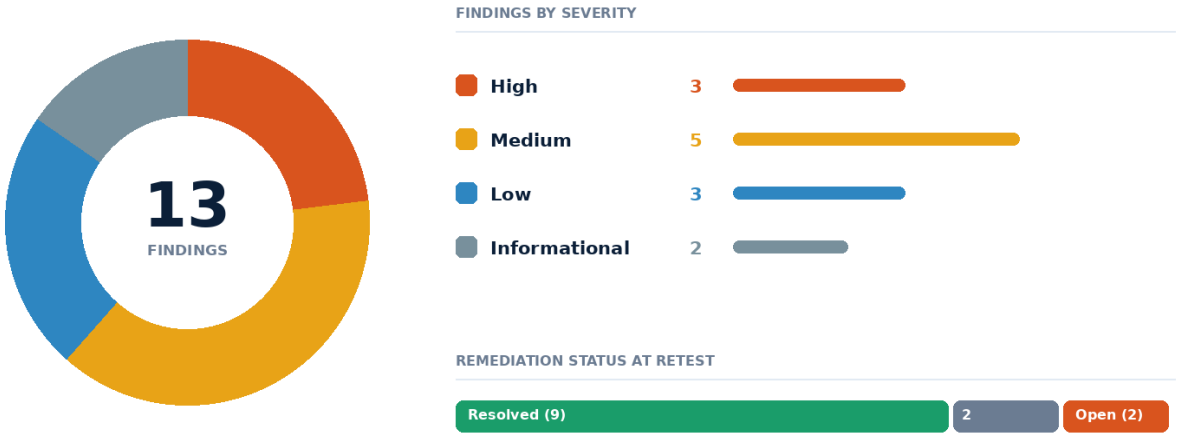


Figure 3.1 — Findings by severity and remediation status at retest.

3.1 What we found

iOS gives an application strong defaults. The sandbox is tight, the Keychain is hardware-backed on every supported device, and data protection classes are applied by the system unless an application opts out. Much of what went wrong on Android is simply not possible here.

The pattern in this report is narrower and subtler: the application ***asks the platform a question and then acts on the answer itself***, rather than letting the platform enforce the outcome. Face ID is asked whether the user is present, and the app decides what to unlock. The Keychain is asked to store a token, and the app chooses an accessibility class that makes the hardware protection irrelevant. Three findings share that shape.

Principal business risks

RISK	HOW IT ARISES	EXPOSURE
Account access from a device in hand	The biometric gate is a boolean the app acts on, and the session Keychain item is readable after first unlock (IOS-01, IOS-02).	**Session reachable without the passcode**

RISK	HOW IT ARISES	EXPOSURE
Customer data recoverable from the container	The Core Data store carries NSFileProtectionNone and is included in unencrypted local backups (IOS-02).	**1,8[redacted] records**
Traffic interception despite pinning	Two jailbreak checks and one pinned anchor, none re-verified server-side (IOS-03).	**Full plaintext of authenticated traffic**
Actions triggered from outside the app	The custom URL scheme executes actions from any caller with no origin check or confirmation (IOS-05).	**Document shared outside the tenant**
Backend trusts the client	The same unvalidated attestation and entitlement pattern found on Android (IOS-04).	**Instrumented clients accepted**

3.2 What was done well

- The binary ships with the expected protections in place: position-independent execution, stack canaries and automatic reference counting, with App Store encryption applied.
- App Transport Security is enabled with a single, documented exception, and no cleartext traffic was observed at any point.
- Cryptographic use is appropriate throughout — platform primitives via CryptoKit, no custom algorithms and no static initialisation vectors.
- Payment flows are delegated to the processor's SDK, so no primary account number is handled by, or recoverable from, the application.
- Authentication against the backend is sound; no bypass was achievable at the API, and token lifetimes behave as documented.
- The team fixed the Keychain accessibility class within three working days of notification, ahead of the formal report.

3.3 Recommended priorities

1. Bind biometric authentication to the Keychain item itself using an access control with biometryCurrentSet, so that Face ID releases the credential rather than merely returning a value the application trusts.
2. Set the Keychain accessibility class to kSecAttrAccessibleWhenUnlockedThisDeviceOnly for session material, and apply complete file protection to the local data store.
3. Exclude the container from unencrypted backups, and verify the exclusion against an actual backup rather than the entitlement.
4. Validate device attestation and entitlements server-side, so that defeating a client-side control is no longer sufficient.
5. Require origin validation and user confirmation for every action reachable through a URL scheme or universal link.

4. Compliance and Regulatory Alignment

Mobile clients sit inside the same compliance obligations as the rest of the platform. Where the application handles regulated data, assessors increasingly ask for platform-specific testing evidence rather than accepting a web application test as coverage for both mobile clients.

4.1 Framework coverage

FRAMEWORK	OBLIGATION ADDRESSED	FINDINGS AFFECTING	POSITION AT ISSUE
ISO/IEC 27001:2022	Annex A 8.8 technical vulnerability management, A 8.26 application security requirements, A 8.28 secure coding, A 8.29 security testing in development and acceptance	All findings	**Evidence provided**
SOC 2 (TSC 2017)	CC6.1 logical access, CC6.6 protection against external threats, CC6.7 data in transit and at rest, CC7.1 vulnerability identification	IOS-01, IOS-02, IOS-03, IOS-04	**Evidence provided**
HIPAA Security Rule	§164.312(a)(1) access control, §164.312(a)(2)(iv) encryption at rest, §164.312(d) person or entity authentication, §164.308(a)(8) periodic technical evaluation	IOS-01, IOS-02, IOS-04, IOS-06	**Findings affecting ePHI paths remediated**
PCI DSS v4.0.1	Req. 6.2.4 secure application development, 6.3.1 vulnerability identification, 11.4.2/11.4.3 penetration testing where the app participates in a payment flow	IOS-02, IOS-03, IOS-08	**Payment path delegated — no PAN handled**
GDPR	Art. 25 data protection by design, Art. 32 security of processing including encryption of personal data at rest	IOS-02, IOS-06, IOS-11	**Art. 32(1)(a) and 32(1)(d) evidence**
App Store review	Guideline 5.1 data collection and storage, and the App Privacy declarations for data linked to the user	IOS-06, IOS-11	**Privacy declaration review recommended**

One backend, two clients, two reports

IOS-04 and the Android finding AND-04 are the same server-side weakness reached from different clients. Where an organisation runs both platforms, Cetonix tests each client separately and cross-references shared backend findings, so a fix is verified from both directions rather than assumed.

4.2 Control mapping by finding

ID	OWASP MASVS	ISO 27001:2022	SOC 2	HIPAA
IOS-01	MASVS-AUTH-2	A 8.5, A 8.26	CC6.1	§164.312(d)
IOS-02	MASVS-STORAGE-1	A 8.24, A 8.26	CC6.1, CC6.7	§164.312(a)(2)(iv)
IOS-03	MASVS-NETWORK-1	A 8.24	CC6.7	§164.312(e)(1)
IOS-04	MASVS-RESILIENCE-1	A 8.26	CC6.1	§164.312(d)
IOS-05	MASVS-PLATFORM-3	A 8.26	CC6.6	—
IOS-06	MASVS-STORAGE-2	A 8.10	CC6.7	§164.312(a)(2)(iv)
IOS-07	MASVS-PLATFORM-2	A 8.26	CC6.6	—
IOS-08	MASVS-CRYPTO-2	A 8.24	CC6.1	—
IOS-09	MASVS-NETWORK-2	A 8.24	CC6.7	§164.312(e)(1)
IOS-10	MASVS-CODE-4	A 8.8	CC7.1	—
IOS-11	MASVS-PRIVACY-1	A 5.34	CC6.7	§164.312(b)
IOS-12	MASVS-CODE-2	A 8.8	CC7.1	—
IOS-13	MASVS-PLATFORM-1	A 8.9	CC6.6	—

5. Scope and Rules of Engagement

Scope, authorisation and constraints were agreed in writing before testing began. The signed rules of engagement are held with the engagement record and are available to assessors on request.

5.1 In-scope assets

ASSET	TYPE	ENVIRONMENT	TEST DEPTH
com.acme.workspace	iOS app	TestFlight build	Static, dynamic, runtime and IPC — full MASVS assessment
IPA (supplied artefact)	Binary	Supplied by client	Binary analysis, secret scanning, entitlement and plist review
api.acme-corp.example	Backend REST API	Production	Authenticated testing of the endpoints the app consumes
acmeworkspace:// scheme	IPC surface	On device	Scheme and universal link abuse testing
Share extension	App extension	On device	Shared container and data handling review
Test devices	Hardware	Cetonix lab	iPhone (stock, iOS 26) and jailbroken equivalent

5.2 Explicitly out of scope

EXCLUDED	REASON
Denial-of-service and volumetric testing	Excluded by agreement; production backend serving live customers
Apple platform infrastructure	Operated by a third party; no authorisation to test
Payment processor SDK internals	Third-party component; tested only at the integration boundary
Secure Enclave and hardware attacks	Outside the agreed scope and the assessed threat model
Source code review	Not commissioned here; recommended as a follow-on for the storage and authentication layers
Android application	Assessed separately under engagement CTX-PT-2026-0431

5.3 Test accounts, devices and roles

ROLE	ACCOUNTS	DEVICE	PURPOSE
Unauthenticated	—	Stock + jailbroken	Pre-login surface, URL schemes, universal links
Standard member	2	Stock + jailbroken	Baseline permissions; container

ROLE	ACCOUNTS	DEVICE	PURPOSE
			and traffic review
Workspace admin	2	Stock + jailbroken	Elevated functionality and integration settings
Second application	n/a	Stock	Custom app used to test scheme handling and shared containers

5.4 Rules of engagement

Testing window	09:00–21:00 client local time, Monday to Friday. Backend testing rate-limited to 5 requests per second.
Device policy	All testing performed on Cetonix-owned handsets in the lab. No client-owned or employee device was touched.
Data handling	Where the container exposed customer records, rows were counted and classified on the device. Nothing was exported.
Destructive actions	No deletion or modification of production records. Write operations limited to objects owned by the test accounts.
Critical escalation	Findings of Critical severity reported to the client's named contact within two hours of confirmation. None arose in this engagement.
Identification	All backend traffic carried the header X-Cetonix-Assessment: CTX-PT-2026-0437 so the client could distinguish it from genuine activity.
Stop conditions	Testing halts immediately on evidence of prior compromise, on production instability, or on client instruction.

Scope limitations affecting coverage

Testing covered the TestFlight build supplied on 14 August 2026. Because App Store builds are re-signed and encrypted by Apple, findings dependent on binary contents should be re-verified against the shipping build. No other constraint materially limited coverage.

6. Methodology

The engagement followed the OWASP Mobile Application Security Verification Standard v2.1, using the test procedures of the Mobile Application Security Testing Guide, within the phase structure of PTES and NIST SP 800-115. Automated tooling was used for coverage and discovery only; every finding was manually reproduced before it was written up.

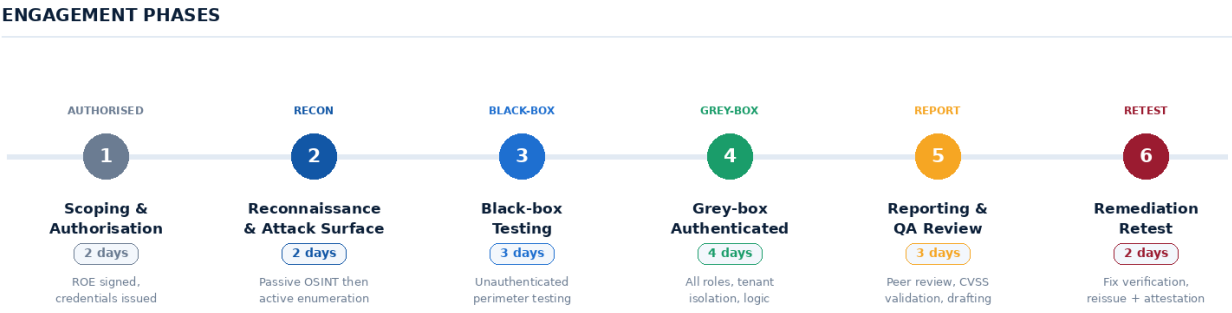


Figure 6.1 — Engagement phases and duration.

6.1 Phase detail

1. Scoping and authorisation	Artefact receipt, account provisioning, device preparation and rules of engagement signed by both parties. No traffic is sent to client systems before this is complete.
2. Reconnaissance and static analysis	Binary analysis, entitlement and Info.plist review, URL scheme and universal link enumeration, secret scanning, third-party SDK review and App Transport Security review. Performed entirely in the Cetonix lab.
3. Unauthenticated device testing	Everything reachable before login or from outside the app: URL schemes, universal links, app extensions, shared containers, pasteboard and pre-authentication storage.
4. Authenticated and runtime testing	Credentialed testing across roles on stock and jailbroken devices: Keychain and container contents, traffic interception, runtime instrumentation, biometric and integrity control strength, and backend authorisation.
5. Reporting and QA	Independent peer review by a reviewer who did not perform the testing, CVSS vector validation, and drafting. One finding was re-scored during review.
6. Remediation retest	Verification of the client's fixes against a new build, re-issue of the report, and issue of the attestation letter.

6.2 Coverage against OWASP MASVS

MASVS CONTROL GROUP	TESTED	OUTCOME
MASVS-STORAGE — data at rest	**Yes**	IOS-02, IOS-06 — principal weakness area
MASVS-CRYPTO — cryptography	**Yes**	IOS-08; algorithm and mode selection otherwise sound
MASVS-AUTH — authentication	**Yes**	IOS-01; backend enforcement correct

MASVS CONTROL GROUP	TESTED	OUTCOME
MASVS-NETWORK — communication	Yes	IOS-03, IOS-09; TLS configuration otherwise correct
MASVS-PLATFORM — platform interaction	Yes	IOS-05, IOS-07, IOS-13
MASVS-CODE — code quality	Yes	IOS-10, IOS-12; no memory-safety issues identified
MASVS-RESILIENCE — anti-tampering	Yes	IOS-04; all client-side controls bypassed
MASVS-PRIVACY — data minimisation	Yes	IOS-11; logging and telemetry reviewed
Backend authorisation (supporting)	Yes	IOS-04; API endpoints consumed by the app exercised across roles

7. Reconnaissance and Attack Surface

The attacker starts with the application, because anyone can download it. Reconnaissance therefore asks two questions: what does the binary disclose, and what does the installed application expose to the rest of the device and to anything that can open a link.

7.1 Static analysis

Static analysis — binary review and data-at-rest inspection

```
$ cetonix-mobile static --ipa ACMEWorkspace.ipa --masvs --profile ios

[*] Bundle         com.acme.workspace   build 8[redacted]   minimum iOS 16.0
[*] Protections    PIE ✓   stack canaries ✓   ARC ✓   encrypted ✓
[*] Symbols        Objective-C class metadata recoverable from binary

[!] SECRET     API key literal in __TEXT/__cstring        [redacted]
[!] SECRET     Analytics write key in Info.plist          [redacted]

[!] KEYCHAIN   kSecAttrAccessibleAfterFirstUnlock on session item
[!] KEYCHAIN   kSecAccessControlBiometryCurrentSet not set
[!] STORAGE    Core Data store unencrypted · NSFileProtectionNone
[!] SNAPSHOT   Account screen captured to snapshot cache on background
[!] ATS        NSExceptionAllowsInsecureHTTPLoads for [redacted]   domain

[=] 11 MASVS control failures · 2 hardcoded secrets · 5 escalated for manual testing
```

Evidence IOS-RECON-01 · Static phase — performed on the supplied build artefact in the Cetonix lab

Evidence IOS-RECON-01 — Binary and data-at-rest analysis of the supplied build. Secrets and domains masked in this copy.

7.2 Discovered attack surface

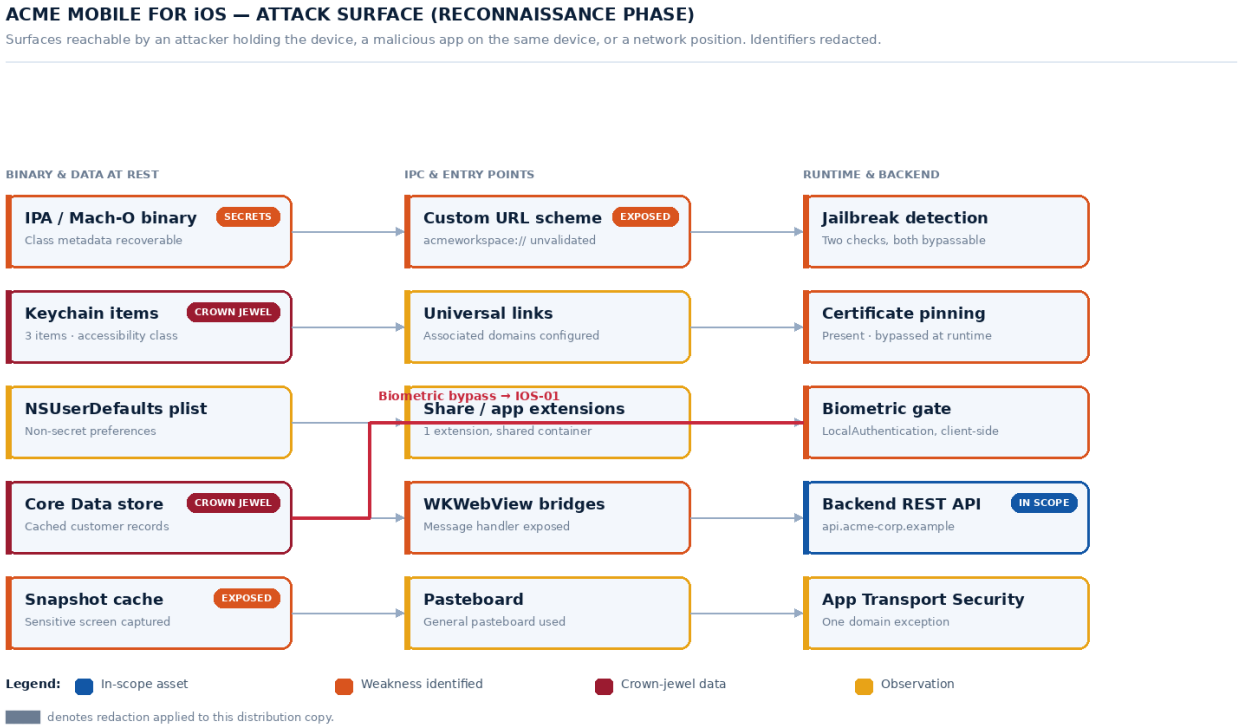


Figure 7.1 — iOS attack surface with the demonstrated attack path overlaid.

7.3 What the binary and container disclose

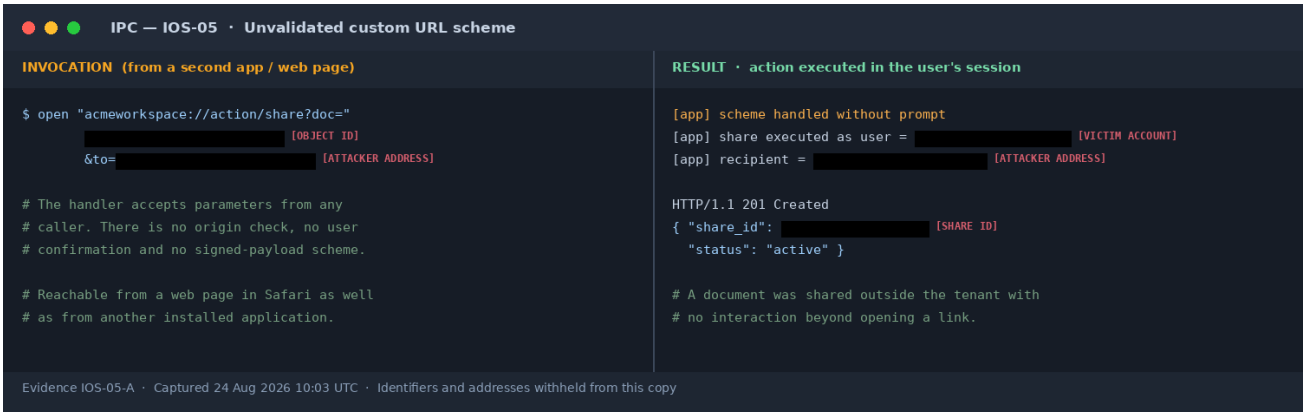
ITEM	STATUS	ASSESSMENT
Binary protections	**Correct**	PIE, stack canaries, ARC and App Store encryption all present. Nothing to raise here.
Class metadata	**Recoverable**	Objective-C class and selector names are readable, which maps the application's internal structure. Expected for the language; worth noting because it accelerates everything that follows.
Hardcoded secrets	**2 found**	An API key in the binary's string table and an analytics write key in Info.plist — raised as IOS-08.
Keychain accessibility	**Weak class**	Session material stored with kSecAttrAccessibleAfterFirstUnlock and no access control, so it is readable while the device is locked — raised as IOS-02.
File protection	**None**	The Core Data store and its journal carry NSFileProtectionNone, opting out of the platform's at-rest encryption — raised as IOS-02.
URL schemes	**1 declared**	acmeworkspace:// is declared and handled without origin validation — raised as IOS-05.
ATS configuration	**1 exception**	One domain permits insecure loads. The domain is a third-party content host — raised as IOS-09.
Third-party SDKs	**7 present**	Analytics, crash reporting and payment SDKs. One analytics SDK receives screen names disclosing customer identifiers — raised as IOS-11.

8. Unauthenticated and IPC Testing

This phase asks what an attacker achieves without the victim's credentials: from a second application, from a web page that can open a link, from an app extension, or from brief possession of an unlocked handset.

8.1 URL scheme and universal link testing

The application declares a custom URL scheme and a set of associated domains. Both were exercised from a second application and from a web page rendered in Safari.



Evidence IOS-05-A — An action executed in the user's session from an externally supplied link. Identifiers masked.

8.2 Results

TEST AREA	RESULT	OBSERVATION
Custom URL scheme handling	**Fail**	Actions executed from any caller with no origin check and no user confirmation — IOS-05
Universal links	**Pass**	Associated domains correctly configured and verified; no hijack achievable
App extension / shared container	**Fail**	The share extension writes unprotected files into the shared container — IOS-07
Pasteboard	**Fail**	Document references placed on the general pasteboard without expiry — IOS-07
Snapshot cache	**Fail**	The account screen is captured on backgrounding and persists in the container — IOS-06
Pre-authentication storage	**Pass**	No sensitive material written before login
Local backup extraction	**Fail**	Container included in unencrypted local backups — IOS-02
WKWebView message handlers	**Observed**	A handler is exposed; all content loaded during testing originated from ACME over pinned HTTPS — IOS-13

8.3 Outcome of the unauthenticated phase

Nothing in this phase gave up the user's session on its own — a meaningful difference from the Android assessment, and a credit to the platform's sandbox. What it did establish is that the container is not a private place once the device is out of the user's hands, and that the application will act on instructions arriving from outside itself.

9. Authenticated and Runtime Testing

The authenticated phase ran across three role levels on both stock and jailbroken handsets. Its purpose is to establish what the application stores, what it sends, and how much of its security depends on decisions made on hardware the attacker controls.

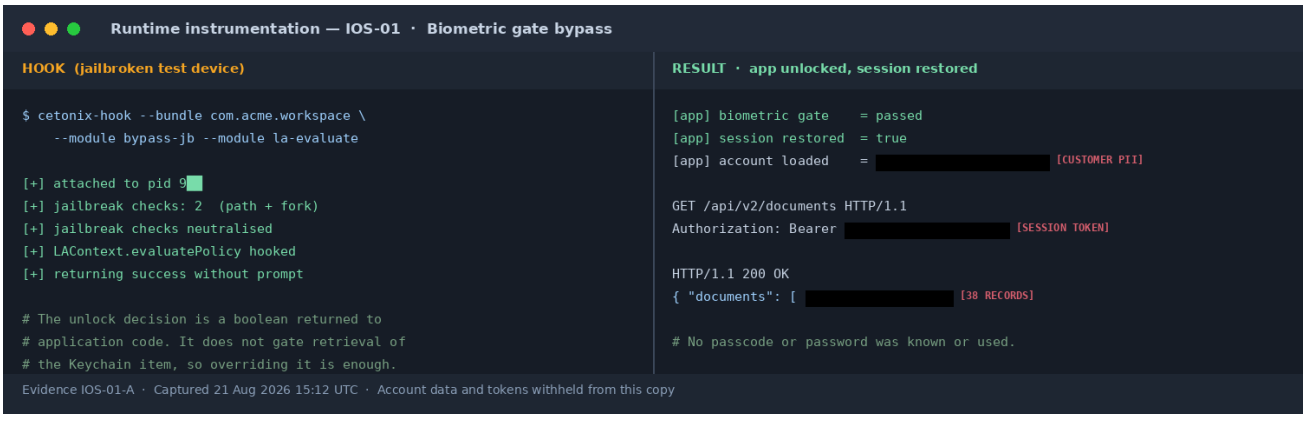
9.1 Data at rest

Evidence IOS-02-A — Keychain items and container contents recovered in the lab. Credential material and customer records masked.

9.2 The biometric gate

Face ID is used to unlock the application on launch. The implementation calls LocalAuthentication, receives a boolean, and — if it is true — reads the session token from the Keychain and restores the session. The Keychain item itself is not protected by an access control, so the biometric result is advisory.

Hooking the evaluation call to return success was sufficient to unlock the application and restore the victim's session, without knowing the device passcode or the account password.



Evidence IOS-01-A — The biometric gate defeated and the session restored. Account data and tokens masked.

9.3 Runtime control strength

CONTROL	EFFORT TO BYPASS	ASSESSMENT
Jailbreak detection	**~12 minutes**	Two checks — file path and fork behaviour — both defeated with standard modules. No secondary check and no server-side signal
Certificate pinning	**~15 minutes**	One pinned anchor hooked at the platform layer. No fallback validation and no telemetry on failure — IOS-03
Biometric gate	**~5 minutes**	A single hooked evaluation call. The Keychain item is not bound to biometry, so the gate is advisory — IOS-01
Anti-debugging	**Not present**	No ptrace or sysctl checks; the process attached without resistance
Server-side attestation	**Not enforced**	DeviceCheck is not used and no attestation is validated at the backend — IOS-04

9.4 Backend behaviour under an instrumented client

- Client-asserted entitlement flags were honoured, exactly as in the Android assessment. A premium-only endpoint executed for an account without the entitlement — IOS-04.
- No device attestation is requested or validated. DeviceCheck and App Attest are both available on the supported platform versions and neither is in use.
- Backend authorisation on customer data was correct. Objects belonging to other tenants were refused, and no cross-tenant access was achievable from the mobile client.
- Rate limits were applied per device identifier, which an instrumented client controls, rather than per account.

9.5 Logging and detectability

ACTIVITY	LOGGED	COMMENT
Authentication from a new device	**Yes**	Recorded with device identifier and source address
Jailbreak signal	**No**	The client detects a jailbroken device but reports nothing to the backend, so the signal is lost
Biometric gate failure or bypass	**No**	Not reported; the gate produces no server-side record at all
Pinning failure	**No**	The connection fails locally and nothing is sent
Anomalous API usage from a client	**Partial**	Volume is recorded; no behavioural baseline exists to alert on

The detectability gap

As on Android, the application knows things the backend would find useful and keeps them to itself. A jailbroken device, a failed pin and a bypassed biometric gate are all signals the client can already produce; none of them currently reaches anyone who could act on them.

10. Use of Artificial Intelligence in this Engagement

Cetonix discloses the role of AI tooling in every engagement, because a client buying assurance is entitled to know what was assessed by a person and what was not. The position is simple: AI accelerates the analyst, and never substitutes for them.

Enterprise models only — your data is never used for training

All AI assistance runs exclusively on enterprise-tier model services procured under commercial agreements that contractually prohibit the use of customer or engagement data for model training, with zero data retention enabled where the provider offers it. Consumer and free-tier AI services are prohibited by Cetonix policy and are blocked on engagement systems. Nothing you disclose to us — code, credentials, captures, findings or this report — enters a training corpus.

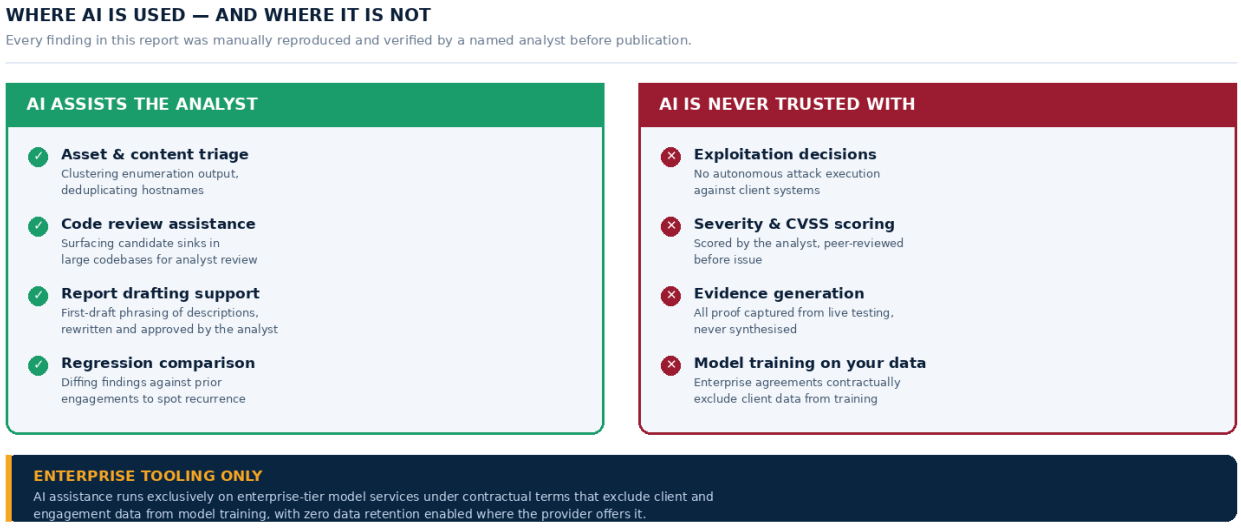


Figure 10.1 — Division between AI-assisted and analyst-only activity.

10.1 Where AI was used in this engagement

ACTIVITY	HOW AI WAS USED	HUMAN CONTROL
Binary triage	Grouping recovered class and selector metadata to surface candidate authentication, storage and crypto code	Analyst read every candidate class; nothing was accepted as a finding on the model's say-so
Control checklist assembly	Cross-referencing the app's entitlements and capabilities against the MASVS control set	All applicable controls were tested manually regardless of the generated checklist
String and secret candidates	Flagging high-entropy strings in the binary and Info.plist as possible secrets	Each candidate was validated against the live backend by the analyst before being reported
Report drafting	First-draft phrasing of finding descriptions and remediation text from analyst notes	Rewritten and approved by the lead consultant; technical content originates from the analyst

ACTIVITY	HOW AI WAS USED	HUMAN CONTROL
Cross-platform comparison	Diffing these findings against the Android assessment to identify shared backend weaknesses	Analyst validated each match; two shared backend issues confirmed and noted in section 15

10.2 Where AI was not used

Exploitation	No autonomous agent was permitted to interact with client systems. Every request that reached the client's systems was issued by, or under the direct control of, a named analyst.
Severity and CVSS scoring	All ratings were assigned by the analyst and independently validated by the QA reviewer. No score in this report was generated by a model.
Evidence	All evidence is captured from live testing. Nothing in section 13 is reconstructed, illustrative or synthesised.
Finding determination	No finding was published on the basis of a model's assertion. Reproduction by a human analyst is a precondition of inclusion.

10.3 Model provenance and data handling

The controls below are contractual and technical, not aspirational. They are available for review during vendor due diligence, and they apply to every engagement Cetonix performs.

Service tier	Enterprise or business-tier model services only, procured under a signed commercial agreement. Consumer and free-tier AI services are prohibited by policy and blocked on engagement systems.
Training exclusion	Every agreement contractually excludes customer content — prompts, files, outputs and engagement material — from model training, fine-tuning and human review for product improvement.
Retention	Zero data retention is enabled where the provider offers it. Where it is not, retention is limited to the provider's minimum abuse-monitoring window and no client identifiers are submitted.
Data minimisation	Client credentials, captured traffic, customer records and personal data are never submitted to an AI service. Assistance operates on analyst notes and Cetonix-side metadata.
Residency and access	Requests are issued from Cetonix-controlled accounts with named-user access, audit logging and no third-party plugin or connector access to engagement material.
Evidence on request	The relevant provider terms and the internal AI usage policy can be supplied to the client's procurement or security function on request.

Our commitment on AI

If a finding appears in a Cetonix report, a named analyst reproduced it, scored it and stands behind it. AI changes how quickly we get to the interesting requests; it does not change who decides what is true, and it never learns from your data.

11. Findings Summary

ID	FINDING	SEVERITY	CVSS	CWE	STATUS
IOS-01	Biometric gate decided in the client, not bound to the Keychain	High	7.6	CWE-287	Resolved
IOS-02	Keychain accessibility class and unprotected local store	High	7.4	CWE-311	Resolved
IOS-03	Certificate pinning and jailbreak detection bypassable	High	7.1	CWE-295	Resolved
IOS-04	Device attestation and entitlements not validated server-side	Medium	6.8	CWE-602	Resolved
IOS-05	Custom URL scheme executes actions without origin validation	Medium	6.1	CWE-939	Resolved
IOS-06	Sensitive screens captured to the snapshot cache	Medium	5.5	CWE-200	Resolved
IOS-07	Shared container and pasteboard leak authenticated content	Medium	5.0	CWE-200	Resolved
IOS-08	Hardcoded API and analytics keys in the binary	Medium	4.8	CWE-798	Resolved
IOS-09	App Transport Security exception for a third-party domain	Low	3.7	CWE-319	Resolved
IOS-10	Verbose logging retained in the shipped build	Low	3.3	CWE-532	Open
IOS-11	Customer identifiers sent to a third-party analytics SDK	Low	3.1	CWE-359	Accepted
IOS-12	Outdated bundled framework with no reachable sink	Info	—	CWE-1104	Open
IOS-13	WKWebView message handler exposed to loaded content	Info	—	CWE-749	Accepted

11.1 Risk positioning

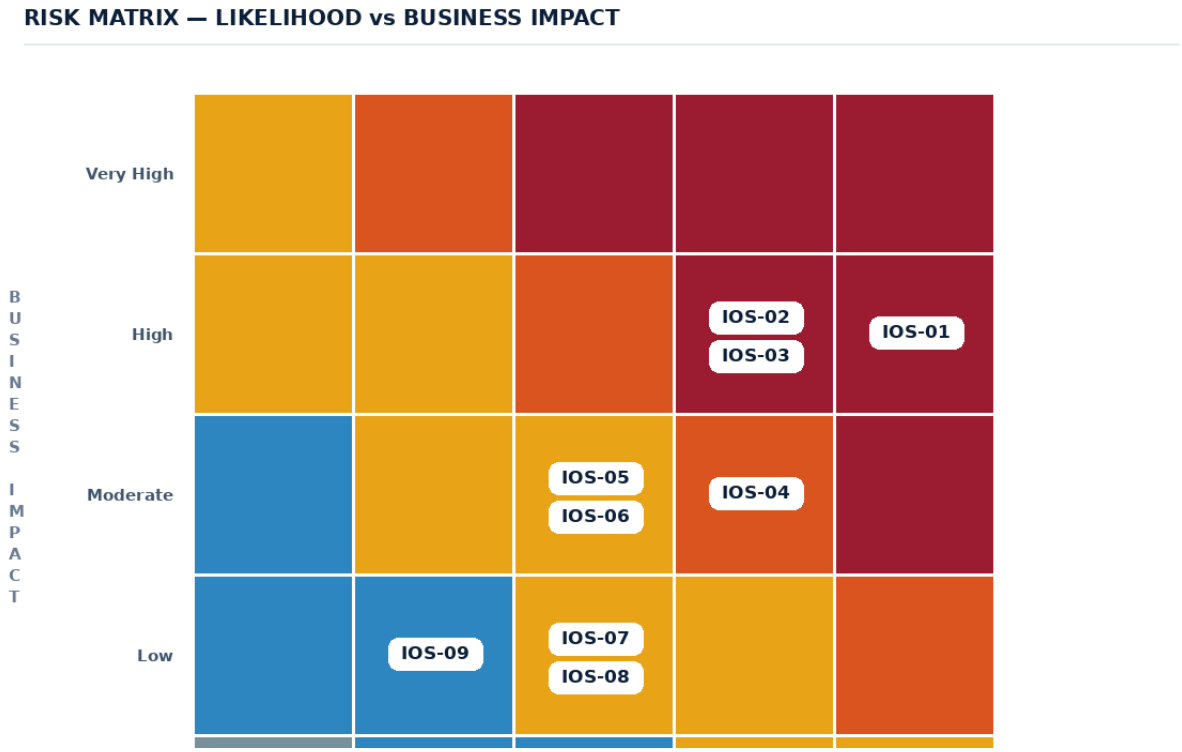


Figure 11.1 — Findings plotted by likelihood of exploitation against business impact.

12. Attack Narrative

No single finding here is critical, and that is worth stating plainly. The sequence below is what happens when several medium-weight weaknesses line up behind one realistic event: someone else ends up holding the phone.

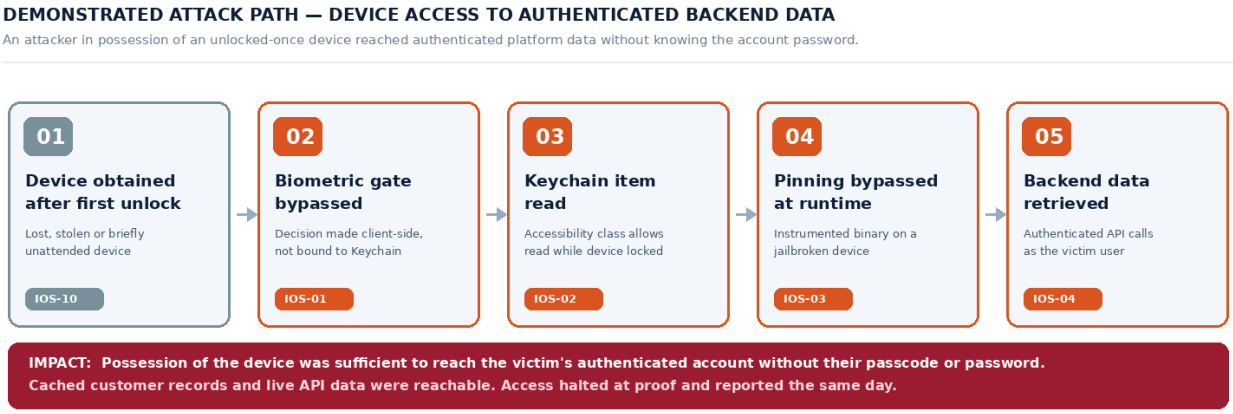


Figure 12.1 — Demonstrated attack path from device possession to authenticated backend data.

12.1 Narrative

1. **Entry.** A device that has been unlocked at least once since boot comes into the attacker's possession — lost, stolen, or briefly unattended. The passcode is not known.
2. **Gate bypass.** The application's Face ID gate is a boolean the app acts on. Hooking the evaluation call returned success and the application unlocked, restoring the previous session (IOS-01).
3. **Credential read.** The session token is held in the Keychain with an accessibility class that permits reads after first unlock and no biometric access control, so the hardware protection never engaged (IOS-02).
4. **Control bypass.** Jailbreak detection and the single pinned anchor were defeated with standard modules, exposing the full plaintext of authenticated traffic (IOS-03).
5. **Objective.** Authenticated backend calls were made as the victim. No attestation is requested or validated, so the server could not distinguish the instrumented client from the genuine application (IOS-04).

> **Elapsed time from taking possession of the device to authenticated backend access: 1 hour 12 minutes**
> The account password was never known and the device passcode was never entered. Every control that should have stopped this exists in the application already — each was simply used as an interface affordance rather than as a boundary.

12.2 What would have broken the chain

- A Keychain access control requiring biometryCurrentSet would have stopped steps 2 and 3 together: the token would not be released, whatever the application's own code believed.

- Complete file protection on the local store would have prevented the cached customer records from being readable in the same scenario.
- Server-side attestation validation would have stopped step 5 independently of every other fix.
- Forwarding the jailbreak and pinning signals the client already produces would not have prevented the chain, but would have made it visible.

13. Detailed Findings

Each finding records what was found, how it was proved, what it means for the business, and what to do about it. Evidence is reproduced as captured, with sensitive values masked according to the key in section 2.

IOS-01 Biometric gate decided in the client, not bound to the Keychain

HIGH
CVSS 7.6

CVSS v3.1	AV:P/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:N	**CWE**	CWE-287 · CWE-603
Affected asset	com.acme.workspace — launch authentication	**OWASP**	MASVS-AUTH-2 · M1:2024
Status	**Resolved — verified 2 Sep 2026**	**Compliance**	ISO A 8.5 · SOC 2 CC6.1 · HIPAA §164.312(d)
References	OWASP MASVS-AUTH-2 · OWASP MASTG-TEST-0018 (testing biometric authentication) · OWASP Mobile Top 10 M1:2024 Improper Credential Usage · CWE-603 Use of Client-Side Authentication		

Description

The application gates launch behind Face ID. It calls LocalAuthentication's evaluation method, receives a boolean, and on success reads the session token from the Keychain and restores the user's session.

The Keychain item holding that token carries no access control object. It is therefore readable by the application at any time, and the biometric result is a decision the application makes about its own behaviour rather than a condition the platform enforces on the credential.

Hooking the evaluation call to return success unlocked the application and restored the victim's session on a jailbroken device, with no knowledge of the device passcode or the account password.

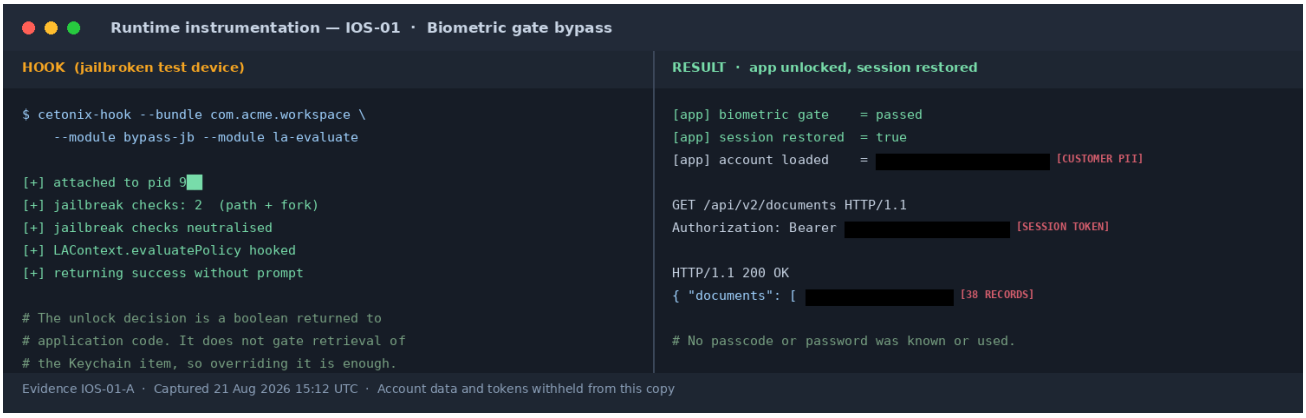
Business impact

Possession of a device that has been unlocked once since boot is sufficient to reach the victim's authenticated account. This is the scenario users believe Face ID protects them from, and it is the scenario the current implementation does not cover.

The finding is the entry point for the chain in section 12: once the session is restored, everything the account can reach is reachable.

For ACME's healthcare and financial customers, the assurance that a lost device does not equal a compromised account is a control their own auditors ask about directly.

Evidence



Evidence IOS-01-A — The biometric evaluation hooked and the session restored. Tokens and account data masked.

Reproduction

1. Sign in to the application on a jailbroken test device and background it so the biometric gate engages on next launch.
2. Attach a dynamic instrumentation framework to the application process.
3. Hook the LocalAuthentication evaluation call and return success unconditionally:
   `LAContext.evaluatePolicy → true`
4. Relaunch the application. The gate is satisfied without a biometric prompt.
5. Observe the session restored and authenticated API calls succeeding as: [redacted]
   [REDACTED — VICTIM ACCOUNT]

Remediation

- Protect the session Keychain item with an access control object created with biometryCurrentSet, so that the platform — not the application — releases the credential only on a successful biometric match.
- Use biometryCurrentSet rather than biometryAny so that enrolling a new face or fingerprint invalidates the item, which is what defeats an attacker who can add their own biometric.
- Require the device passcode as the fallback, not an application-level password or a bypass path.
- Do not treat the evaluation result as an authorisation decision anywhere in application code; the only thing that should depend on it is the platform releasing the item.
- Re-authenticate against the backend after a configurable period of inactivity, so that a restored local session is not indefinitely valid.

> **Let the platform enforce, don't ask it for an opinion**
> IOS-01 and IOS-02 are the same mistake at two layers: using a platform security primitive as a signal the application interprets, rather than as a boundary the platform applies. Binding the Keychain item to biometry fixes both at once.

IOS-02 Keychain accessibility class and unprotected local store

HIGH CVSS 7.4

CVSS v3.1	AV:P/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N	**CWE**	CWE-311 · CWE-922
Affected asset	com.acme.workspace — Keychain and app container	**OWASP**	MASVS-STORAGE-1 · M9:2024
Status	**Resolved — verified 2 Sep 2026**	**Compliance**	ISO A 8.24 · SOC 2 CC6.7 · HIPAA §164.312(a)(2)(iv) · GDPR Art. 32
References	OWASP MASVS-STORAGE-1 · OWASP MASTG-TEST-0052 (testing Keychain item accessibility) · OWASP Mobile Top 10 M9:2024 Insecure Data Storage · CWE-311 Missing Encryption of Sensitive Data		

Description

Session and refresh tokens are stored in the Keychain with kSecAttrAccessibleAfterFirstUnlock, which permits reads whenever the device has been unlocked once since boot — including while it is locked and in the owner's pocket. No access control object is attached.

Separately, the Core Data store and its write-ahead journal carry NSFileProtectionNone, opting out of the platform's file-level encryption. The store held 1,8[redacted] cached customer records at the time of testing.

The container is also included in unencrypted local backups, so the same material is recoverable without touching the device beyond connecting it.

Business impact

The two strongest protections iOS offers — hardware-backed Keychain access control and file data protection — are both present on the device and both effectively disabled by configuration.

A lost or stolen handset gives up the user's session and a substantial extract of the customer data they had viewed. For ACME's healthcare customers this is a disclosure of ePHI; for EU customers it is a personal data breach under GDPR.

Because the weakness is a class and attribute choice rather than a code defect, it is invisible in code review and easy to reintroduce.

Evidence

Evidence IOS-02-A — Keychain items and container contents recovered in the lab. Values masked.

Reproduction

1. Sign in on a test device and browse several customer records so the local store is populated.
2. Inspect the Keychain item attributes for the application's session entries and observe the accessibility class and the absence of an access control object.
3. Extract the application container to an analysis machine.
4. Open the Core Data store with any SQLite client. No key is required; customer records are readable in full.
5. Create an unencrypted local backup and confirm the same material is present in the archive.

Remediation

- Set kSecAttrAccessibleWhenUnlockedThisDeviceOnly on session material, and attach an access control requiring biometryCurrentSet as described in IOS-01.
- Apply NSFileProtectionComplete to the local store and its journal, and verify the attribute on a real device rather than in the simulator.
- Exclude the container from unencrypted backups by setting the appropriate resource value, and verify against an actual backup.
- Define a retention policy for cached customer data and clear it on logout, on token expiry and after a period of inactivity.
- Add a startup assertion in debug builds that fails loudly if the protection class or accessibility attribute is not what the policy requires.

IOS-03 Certificate pinning and jailbreak detection bypassable

HIGH
CVSS 7.1

CVSS v3.1	AV:N/AC:H/PR:N/UI:R/S:U/C:H/I:H/A:N	**CWE**	CWE-295 · CWE-693
Affected asset	com.acme.workspace — network and integrity layers	**OWASP**	MASVS-NETWORK-1 · M5:2024
Status	**Resolved — verified 2 Sep 2026**	**Compliance**	ISO A 8.24 · SOC 2 CC6.7 · HIPAA §164.312(e)(1)
References	OWASP MASVS-NETWORK-1 · OWASP MASTG-TEST-0020 (testing certificate pinning) · OWASP Mobile Top 10 M5:2024 Insecure Communication · CWE-295 Improper Certificate Validation		

Description

The application pins its backend certificate and performs two jailbreak checks. Both are reasonable instincts and both are implemented in a way that yields to standard tooling: one pinned anchor validated in a single place, and two detection checks of well-known types.

Jailbreak detection was defeated in approximately twelve minutes and pinning in a further fifteen, using publicly available modules requiring no customisation. Neither failure produces a signal to the backend.

Business impact

An attacker with a jailbroken device reads and modifies the full plaintext of authenticated API traffic, including session tokens, customer records and document download URLs.

The practical consequence is that the client can be made to say anything. Combined with IOS-04, the backend has no way to distinguish an instrumented client from the genuine application.

Pinning strength also determines how quickly a researcher or competitor can map the API surface. Fifteen minutes is not a meaningful barrier.

Evidence

Reproduction

1. Prepare a jailbroken test device with a standard dynamic instrumentation framework.
2. Attach to the application process and load a generic jailbreak-detection bypass module. The application proceeds normally.
3. Load a generic pinning bypass module targeting the platform TLS validation path.
4. Route device traffic through an intercepting proxy presenting an untrusted certificate.
5. Observe full plaintext request and response bodies for all authenticated endpoints. No error, alert or backend signal was produced.

Remediation

- Pin multiple anchors — the leaf and a backup key — so that a single hook and a single certificate rotation are both survivable.

- Validate in more than one place and at more than one layer, so that hooking a single method does not defeat the control.
- Report pinning failures and jailbreak verdicts to the backend as telemetry. A client that cannot validate its server is a signal worth having.
- Add anti-debugging checks so that attaching to the process is at least inconvenient, while accepting that these too are cost-raising rather than preventive.
- Combine with server-side attestation validation (IOS-04), which is the control that actually resists client-side tampering.

IOS-04 Device attestation and entitlements not validated server-side

MEDIUM CVSS 6.8

CVSS v3.1	AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:L/A:N	**CWE**	CWE-602 · CWE-807
Affected asset	api.acme-corp.example — all mobile endpoints	**Status**	**Resolved — verified 2 Sep 2026**
Compliance	ISO A 8.26 · SOC 2 CC6.1 · HIPAA §164.312(d)		
References	OWASP MASVS-RESILIENCE-1 · OWASP MASTG-TEST-0043 · OWASP Mobile Top 10 M8:2024 Security Misconfiguration · CWE-602 Client-Side Enforcement of Server-Side Security		

Description

No device attestation is requested or validated. DeviceCheck and App Attest are both available on the supported platform versions and neither is in use, so the backend has no means of distinguishing a genuine application instance from an instrumented one.

The backend also honours client-asserted entitlement flags: a request claiming a premium entitlement was processed for an account that does not hold one. This is the same server-side weakness recorded as AND-04 in the Android assessment.

Business impact

Every client-side control in this report — jailbreak detection, pinning, the biometric gate — is enforced only on the device. Because the server does not verify that it is talking to a genuine, unmodified client, defeating any one of those controls is sufficient. The client is trusted to report on its own integrity and on its own entitlements.

Evidence

```
POST /api/v2/documents/export HTTP/1.1
Host: api.acme-corp.example
Authorization: Bearer [REDACTED — SESSION TOKEN]
X-Client-Entitlement: premium
# no attestation header is sent, requested or expected

HTTP/1.1 200 OK
{ "export_id": [REDACTED — EXPORT ID]
  "tier": "premium" }
# entitlement is not held by the account
```

Remediation

- Adopt App Attest for genuine-client assurance and validate the assertion server-side on sensitive operations.
- Derive entitlements from server-side account state. Never accept an entitlement, tier or role asserted by the client.
- Bind rate limiting to the authenticated account rather than to a device identifier the client controls.

- Log attestation failures and tamper signals as security events, and alert on volume. Note that this fix is shared with the Android client — verify from both.

IOS-05 Custom URL scheme executes actions without origin validation

MEDIUM
CVSS 6.1

CVSS v3.1	AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N	**CWE**	CWE-939
Affected asset	com.acme.workspace — acmeworkspace:// handler	**Status**	**Resolved — verified 2 Sep 2026**
Compliance	ISO A 8.26 · SOC 2 CC6.6		
References	OWASP MASVS-PLATFORM-3 · OWASP MASTG-TEST-0029 (testing deep links) · CWE-939 Improper Authorization in Handler for Custom URL Scheme		

Description

The custom URL scheme handler accepts an action selector and parameters from any caller — a second application, or a web page rendered in Safari — and performs the action in the signed-in user's session with no origin check and no confirmation.

A document share to an externally supplied address was executed from a link, with no interaction beyond opening it.

Business impact

A link in an email, a message or a web page can cause the application to act in the user's session. Because custom schemes are not domain-verified, any application may also register the same scheme and receive links intended for ACME.

Remediation

- Move link handling to verified universal links, which are bound to your domain, and treat the custom scheme as legacy.
- Require explicit user confirmation for any state-changing action reached through a link.
- Validate every parameter server-side and re-check the user's entitlement to the target object after the link resolves.
- Do not accept an action selector from the link; map links to a small, explicit set of safe destinations.

IOS-06 Sensitive screens captured to the snapshot cache

MEDIUM
CVSS 5.5

CVSS v3.1	AV:P/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N	**CWE**	CWE-200
Affected asset	com.acme.workspace — account and document screens	**Status**	**Resolved — verified 2 Sep 2026**
Compliance	ISO A 8.10 · SOC 2 CC6.7 · HIPAA §164.312(a)(2)(iv)		
References	OWASP MASVS-STORAGE-2 · OWASP MASTG-TEST-0014 (testing for sensitive data in backgrounding snapshots) · CWE-200 Exposure of Sensitive Information		

Description

iOS captures a snapshot of the current screen when an application is backgrounded, to animate the app switcher. The application does not obscure sensitive content before that happens, so the account screen — including customer identifiers — is written to the container as an image.

The snapshot persists after the application is closed and is recoverable from the container and from backups.

Business impact

Customer data leaves the protected data model and lands in an image file that inherits the container's file protection — which, per IOS-02, was none. It is also visible in the app switcher to anyone holding the unlocked device.

Remediation

- Overlay a blank or branded view in the scene-will-resign-active callback and remove it on becoming active again.
- Apply the same treatment to any screen displaying customer data, not only the account screen.
- Ensure the snapshot cache inherits complete file protection once IOS-02 is remediated.
- Verify on a device rather than the simulator; snapshot behaviour differs between the two.

IOS-07 Shared container and pasteboard leak authenticated content

MEDIUM
CVSS 5.0

CVSS v3.1	AV:L/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N	**CWE**	CWE-200 · CWE-524
Affected asset	com.acme.workspace — share extension and pasteboard	**Status**	**Resolved — verified 2 Sep 2026**
Compliance	ISO A 8.10 · SOC 2 CC6.7		
References	OWASP MASVS-PLATFORM-2 · OWASP MASTG-TEST-0006 · CWE-524 Use of Cache Containing Sensitive Information		

Description

The share extension writes files into the app group's shared container without file protection, where they persist after the share completes and are reachable by every component in the group.

Document references are placed on the general pasteboard with no expiry and without being marked as sensitive, so they are available to other applications and to Universal Clipboard on the user's other devices.

Business impact

Authenticated content leaves the application's own boundary and lands somewhere the user would not expect — another device, another app's paste buffer, or a file that outlives the action that created it. Individually minor; collectively it widens the surface that IOS-02 already exposes.

Remediation

- Apply complete file protection to shared-container files and delete them when the extension's work finishes.
- Use a local-only pasteboard for sensitive values, mark items sensitive so they are excluded from Universal Clipboard, and set an expiry.
- Review which screens genuinely need a copy action and remove it elsewhere.

IOS-08 Hardcoded API and analytics keys in the binary

MEDIUM
CVSS 4.8

CVSS v3.1	AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N	**CWE**	CWE-798
Affected asset	com.acme.workspace — binary and Info.plist	**Status**	**Resolved — verified 2 Sep 2026**
Compliance	ISO A 8.24 · SOC 2 CC6.1 · PCI DSS 6.2.4		
References	OWASP MASVS-CRYPTO-2 · OWASP MASTG-TEST-0012 (testing for hardcoded secrets) · OWASP Mobile Top 10 M1:2024 · CWE-798 Use of Hard-coded Credentials		

Description

Two secrets were recovered from the supplied build: a third-party API key in the binary's string table, and an analytics write key in Info.plist. Neither is derived at runtime.

App Store encryption protects the binary on disk but not from a process that can read its own memory, so neither key is meaningfully protected on a jailbroken device.

Business impact

Both keys are usable against their respective providers at ACME's expense and under ACME's quota. They should be treated as disclosed from the day the build was distributed.

Remediation

- Rotate both keys.
- Proxy third-party API calls through your own backend so provider keys never reach the device.
- Where a key must be present, fetch it at runtime after authentication and hold it in the Keychain rather than embedding it.
- Add secret scanning to the build pipeline covering the compiled binary and Info.plist, not source alone.

IOS-09 App Transport Security exception for a third-party domain

LOW
CVSS 3.7

CVSS v3.1	AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:L/A:N	**CWE**	CWE-319
Affected asset	com.acme.workspace — Info.plist ATS configuration	**Status**	**Resolved — verified 2 Sep 2026**
Compliance	ISO A 8.24 · SOC 2 CC6.7 · HIPAA §164.312(e)(1)		
References	OWASP MASVS-NETWORK-2 · OWASP MASTG-TEST-0019 · CWE-319 Cleartext Transmission of Sensitive Information		

Description

App Transport Security is enabled with a single exception permitting insecure HTTP loads for one third-party content domain. No authenticated data traverses that domain, and no cleartext traffic to it was observed during testing.

The exception nonetheless removes the platform's transport guarantee for anything that domain serves, including content rendered inside the application.

Business impact

Content loaded over the exempted domain can be modified in transit by anyone on the network path. Where that content is rendered in a WebView with an exposed message handler (IOS-13), the exception and the handler become interesting together.

Remediation

- Ask the third party to serve over HTTPS and remove the exception; this is now a reasonable expectation of any content provider.
- If the exception must remain, scope it as narrowly as the platform allows and never render exempted content in a WebView that exposes a message handler.
- Re-review ATS exceptions at each release; they accumulate quietly.

IOS-10 Verbose logging retained in the shipped build — LOW CVSS 3.3

CVSS v3.1	AV:P/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N	**CWE**	CWE-532
Affected asset	com.acme.workspace — logging	**Status**	**Open — scheduled for the next release**
Compliance	ISO A 8.8 · SOC 2 CC7.1		
References	OWASP MASVS-CODE-4 · OWASP MASTG-TEST-0004 · CWE-532 Insertion of Sensitive Information into Log File		

Description

Debug-level logging remains active in the shipped build. Request paths, response summaries and user identifiers are written to the unified log, with occasional token fragments.

The unified log is not readable by other applications, which limits the impact, but it is included in sysdiagnose bundles and reaches crash-reporting payloads.

Business impact

Log content reaches places the application does not control — diagnostic bundles attached to support tickets, and crash reporting services. Token fragments in particular should never leave the application.

Remediation

- Strip debug logging from release builds through the build configuration, and verify on a release artefact.
- Use the unified logging privacy qualifiers so that dynamic values are redacted by default rather than by remembering to mark them.
- Review what the crash-reporting SDK attaches by default; breadcrumbs frequently contain more than teams expect.

IOS-11 Customer identifiers sent to a third-party analytics SDK

LOW CVSS 3.1

CVSS v3.1	AV:N/AC:H/PR:L/UI:N/S:U/C:L/I:N/A:N	**CWE**	CWE-359
Affected asset	com.acme.workspace — analytics integration	**Status**	**Risk accepted by client**
Compliance	ISO A 5.34 · SOC 2 CC6.7 · GDPR Art. 5, Art. 28 · HIPAA §164.312(b)		
References	OWASP MASVS-PRIVACY-1 · OWASP MASTG-TEST-0208 · GDPR Art. 5(1)(c) · CWE-359 Exposure of Private Personal Information to an Unauthorized Actor		

Description

Screen-view events sent to a third-party analytics provider include screen names constructed from customer record identifiers, and user properties include the signed-in email address. This mirrors AND-12 in the Android assessment.

The provider is a processor under the client's data processing agreement, so this is a data-minimisation and disclosure question rather than an unauthorised transfer.

Business impact

Personal data is shared with a processor beyond what the analytics purpose requires — a GDPR Article 5 minimisation concern, and a disclosure the App Privacy declaration and privacy notice should reflect. For healthcare customers, identifiers associated with clinical screens may constitute ePHI leaving the covered boundary.

Remediation

- Replace identifiers in screen names with stable, non-identifying screen labels.
- Use a pseudonymous analytics identifier rather than the email address as a user property.
- Review the App Privacy declaration and the processing agreement against what is actually transmitted.
- Fix once for both platforms — the same instrumentation exists in the Android client.

IOS-12 Outdated bundled framework with no reachable sink

INFORMATIONAL
CVSS —

A bundled third-party framework is several minor versions behind current and carries a published advisory. The vulnerable code path is not reachable from application code, so no exploitation was achievable and no severity is assigned.

It is recorded because dependency currency is a control in its own right — a future change could make the sink reachable without anyone noticing that a known-vulnerable version is bundled, and enterprise customers increasingly ask for a mobile software bill of materials.

Remediation

- Update the framework as part of routine maintenance.
- Introduce software composition analysis into the mobile pipeline, with a policy distinguishing reachable from unreachable vulnerable code.
- Produce and maintain an SBOM for the mobile artefact alongside the backend one.

References: OWASP MASVS-CODE-2 · OWASP MASTG-TEST-0215 · CWE-1104 Use of Unmaintained Third Party Components

IOS-13 WKWebView message handler exposed to loaded content

INFORMATIONAL
CVSS —

A WKWebView exposes a script message handler to loaded content. All content loaded into this WebView during testing originated from ACME's own domain over pinned HTTPS, so no untrusted content reached the handler and no exploitation was achievable.

It is recorded because the configuration removes the margin for error, and because the ATS exception in IOS-09 means at least one domain in the application's configuration is not transport-protected. If content from that domain were ever rendered here, the handler would be directly reachable by it.

Remediation

- Remove the message handler if the functionality can be achieved through navigation delegate callbacks instead.
- Restrict the WebView to an explicit allow-list of origins and refuse off-domain navigation.
- Validate every message received by the handler as untrusted input, including its type and structure.

References: OWASP MASVS-PLATFORM-1 · OWASP MASTG-TEST-0032 (testing WebViews) · CWE-749 Exposed Dangerous Method or Function

14. Remediation Roadmap

Findings are sequenced by risk reduction per unit of effort rather than by severity alone. The client completed the immediate and short-term actions during the retest window.

14.1 Prioritised actions

ID	ACTION	PRIORITY	EFFORT	OWNER
IOS-01	Bind the session Keychain item to biometryCurrentSet	Immediate	Low	Mobile engineering
IOS-02	Correct the accessibility class; apply file protection	Immediate	Low	Mobile engineering
IOS-03	Multiple pinned anchors; report pinning failures	Short term	Low	Mobile engineering
IOS-04	Adopt App Attest; validate entitlements server-side	Short term	Medium	Backend engineering
—	Forward jailbreak and pinning signals to the backend	Short term	Low	Mobile + backend
IOS-06	Obscure sensitive screens before backgrounding	Short term	Trivial	Mobile engineering
IOS-08	Rotate keys; proxy provider calls through the backend	Short term	Medium	Mobile + backend
IOS-05	Migrate to universal links; confirm state changes	Medium term	Medium	Mobile engineering
IOS-07	Protect shared container files; local-only pasteboard	Medium term	Low	Mobile engineering
IOS-09	Remove the ATS exception	Medium term	Low	Mobile + vendor
IOS-10	Strip debug logging from release builds	Medium term	Low	Mobile engineering
IOS-11	Remove identifiers from analytics payloads	Routine	Low	Product + privacy
IOS-12	Update bundled framework; add SCA and SBOM	Routine	Low	Mobile engineering
IOS-13	Remove or constrain the WebView message handler	Routine	Low	Mobile engineering

14.2 Structural recommendations

The iOS client is close to where it should be. The gap between this report and a clean one is narrower than the finding count suggests, because most items are configuration rather than design.

- **Use platform primitives as boundaries, not signals.** Keychain access control and file protection are enforced by hardware the attacker cannot instrument. Asking Face ID for a boolean is not. This single change of stance resolves IOS-01, IOS-02 and IOS-06.
- **Move the trust boundary to the server.** App Attest plus server-side entitlement derivation makes client-side bypass insufficient, which blunts IOS-03 and IOS-04 together — and the fix is shared with the Android client.
- **Treat links as untrusted input.** Universal links, server-side validation and confirmation for state changes resolve IOS-05 and reduce the phishing surface generally.
- **Give the backend eyes on the device.** The client already knows when it is jailbroken or unable to pin. Forwarding those signals converts silent client-side checks into detection.
- **Assert your own configuration in CI.** Accessibility classes, protection attributes and ATS exceptions are all invisible in code review and easy to regress. A build-time assertion on the release artefact catches them before an assessor does.

15. Retest Results

The client remediated during and after the engagement and a formal retest was performed against build 9[redacted] on 1–2 September 2026. Each finding was re-tested using the original reproduction steps, and each fix was probed for bypass rather than merely confirmed present.

ID	FINDING	RESULT	VERIFICATION
IOS-01	Biometric gate	**Resolved**	Session item now protected by an access control requiring biometryCurrentSet. Hooking the evaluation call no longer releases the credential; the platform refuses the read.
IOS-02	Keychain class and file protection	**Resolved**	Accessibility class changed to WhenUnlockedThisDeviceOnly; complete file protection applied to the store and journal; container excluded from unencrypted backups. All three verified on device.
IOS-03	Pinning and jailbreak detection	**Resolved**	Two anchors pinned, validation at two layers, failures reported to the backend. Bypass required substantially more effort and produced a server-side signal.
IOS-04	Attestation and entitlements	**Resolved**	App Attest adopted and validated server-side; client-asserted entitlements ignored. Verified from both the iOS and Android clients.
IOS-05	URL scheme handling	**Resolved**	Universal links now primary; state-changing actions require confirmation and server-side entitlement re-check.
IOS-06	Snapshot cache	**Resolved**	Sensitive screens obscured on resign-active. Verified by inspecting the snapshot cache after backgrounding.
IOS-07	Shared container and pasteboard	**Resolved**	Shared files protected and removed after use; pasteboard entries marked sensitive and local-only.
IOS-08	Hardcoded keys	**Resolved**	Both keys rotated; the provider call is now proxied through the backend. Re-scanned the new artefact — no literals found.
IOS-09	ATS exception	**Resolved**	Third party now serves over HTTPS; the exception has been removed from Info.plist.
IOS-10	Verbose logging	**Open**	Scheduled for the next release. Cetonix recommends prioritising the removal of token fragments specifically.
IOS-11	Analytics identifiers	**Accepted**	Client accepts the risk for the general product, with a commitment to exclude clinical screens for healthcare customers.

ID	FINDING	RESULT	VERIFICATION
IOS-12	Outdated framework	**Open**	Scheduled in the routine maintenance cycle.
IOS-13	WebView message handler	**Accepted**	Client accepts, with a commitment to review before any third-party content is loaded into this WebView.

Post-retest position

All High findings are resolved, and the two changes that mattered most — binding the Keychain item to biometry and adopting server-side attestation — were implemented properly rather than worked around. Residual risk is limited to two Low and two Informational items, of which two are accepted with compensating controls.

15.1 Findings shared with the Android assessment

Two findings in this report are the same underlying weakness as findings in the Android engagement (CTX-PT-2026-0431), reached from a different client: IOS-04 and AND-04 are one server-side control gap, and IOS-11 and AND-12 are one analytics instrumentation decision. Both were fixed once and verified twice, from each client in turn. Where an organisation ships two mobile clients against one backend, that cross-check is the practical argument for assessing both in the same quarter.

Appendix A — Severity and Scoring Methodology

Severity is assigned from CVSS v3.1 base scores, adjusted for the client's environment and reviewed independently before issue. Where the environmental context materially changes the risk, the adjustment and its reasoning are stated in the finding.

SEVERITY	CVSS	DEFINITION AND EXPECTED RESPONSE
Critical	9.0 – 10.0	Direct compromise of sensitive data or platform integrity, exploitable with little effort and no special access. Notified within two hours of confirmation; remediation expected immediately.
High	7.0 – 8.9	Significant compromise, or a reliable step toward one. Notified within one business day; remediation expected within 30 days.
Medium	4.0 – 6.9	Meaningful weakening of the security posture, typically requiring a precondition or chaining. Remediation expected within 90 days.
Low	0.1 – 3.9	Limited direct impact; contributes to attacker capability or defence in depth. Address in routine maintenance.
Informational	—	No direct security impact at present; recorded for completeness, hygiene or future relevance.

Business impact is assessed separately from technical severity. A technically modest flaw affecting regulated data may be escalated, and a technically severe flaw in an isolated component may be moderated. Every adjustment is recorded in the finding and validated by the QA reviewer.

Appendix B — Tooling

Tools support coverage and discovery. No finding in this report rests on tool output alone; each was reproduced manually before inclusion.

CATEGORY	PURPOSE	NOTES
Binary analysis	Recovering class metadata, strings and load commands from the Mach-O binary	Static review performed in the Cetonix lab on the supplied artefact
Entitlement and plist review	Enumerating capabilities, ATS exceptions and declared URL schemes	Every declared entry point tested manually
Dynamic instrumentation	Hooking runtime methods to test client-side controls	Jailbroken test device; used to assess jailbreak detection, pinning and biometric gating
Intercepting proxy	Inspecting and manipulating backend traffic	Primary manual testing platform for the network phase
Container inspection	Reviewing the app container, Keychain items, caches and	Performed on both jailbroken and stock devices

CATEGORY	PURPOSE	NOTES
	backups	
Custom tooling	Second application and web page used to exercise URL schemes and universal links	Written for this engagement

Appendix C — Entitlement and Entry-Point Inventory

The application's declared capabilities and entry points, as assessed. Third-party domains are redacted in this copy.

ENTRY POINT / CAPABILITY	TYPE	REACHABLE	DISPOSITION
acmeworkspace://	URL scheme	**Any caller**	IOS-05 — confirmation and validation added at retest
Associated domains	Universal links	**Verified**	Correctly configured — no issue
Share extension	App extension	**System**	IOS-07 — container files now protected
Keychain access group	Entitlement	**App + extension**	Shared with the extension — reviewed, appropriate
App group container	Entitlement	**App + extension**	IOS-07 — file protection applied at retest
Background modes	Entitlement	**System**	Remote notifications only — appropriate
[redacted] (ATS exception)	Network	**Removed**	IOS-09 — exception removed at retest
Face ID usage description	Privacy	—	Present and accurate

Appendix D — Glossary

IPA	The packaged iOS application. It can be unpacked and its binary analysed by anyone who obtains it.
Keychain	The iOS system credential store. Its accessibility class determines when an item can be read — the flaw in IOS-02 is choosing a class that permits reads while the device is locked.
MASVS / MASTG	OWASP's Mobile Application Security Verification Standard and the accompanying Testing Guide — the control set and test procedures used throughout this engagement.
LocalAuthentication	The iOS framework providing Face ID and Touch ID. Using it to gate application logic rather than to release a Keychain item is the flaw in IOS-01.
Jailbreak detection	Client-side checks intended to detect a compromised device. Because they run on the device under the attacker's control, they raise cost rather than prevent.
Black-box testing	Testing with no credentials and no prior knowledge, simulating an external attacker.
Grey-box testing	Testing with credentials at each role level, simulating an attacker who has obtained an account.
CVSS	Common Vulnerability Scoring System — an industry-standard method of expressing technical severity as a score and a vector string.

CWE	Common Weakness Enumeration — a catalogue of software weakness types, used to classify the underlying defect.
Rules of engagement	The written agreement defining what may be tested, when, how aggressively, and what is prohibited.

Appendix E — Attestation Letter

The letter below may be shared with auditors, assessors, customers and prospects. It confirms that testing was performed and that findings were remediated, without disclosing technical detail about the platform.

CETONIX

SECURITY ASSURANCE · OFFENSIVE SECURITY PRACTICE

4 September 2026

LETTER OF ATTESTATION — PENETRATION TEST

To whom it may concern,

Cetonix was engaged by ACME Corporation, Inc. to perform an independent penetration test of the ACME Mobile application for iOS (bundle identifier com.acme.workspace) and its supporting backend interfaces. Testing was conducted between 17 and 28 August 2026 under engagement reference CTX-PT-2026-0437.

The assessment comprised static analysis of the supplied build, dynamic and runtime testing on both jailbroken and unmodified devices, inter-process communication testing through URL schemes, universal links and app extensions, and authenticated testing of the backend interfaces the application consumes. The methodology followed the OWASP Mobile Application Security Verification Standard v2.1 and the Mobile Application Security Testing Guide, within the phase structure of PTES and NIST Special Publication 800-115. All findings were manually verified by a named analyst and independently peer-reviewed prior to issue.

Thirteen findings were identified, comprising three of High severity, five of Medium severity, three of Low severity and two of an informational nature. No finding was rated Critical. A remediation retest was performed on 1 and 2 September 2026. All findings rated High have been remediated and independently verified as resolved. Remaining items are of Low or informational severity and are either scheduled for remediation or formally accepted by the client with compensating controls in place.

This engagement provides technical testing evidence supporting ISO/IEC 27001:2022 Annex A 8.8, A 8.26 and A 8.29, SOC 2 Trust Services Criteria CC6.1 and CC7.1, the evaluation requirement at 45 CFR §164.308(a)(8) of the HIPAA Security Rule where the application processes ePHI, and the application-layer testing expectations of PCI DSS v4.0.1 Requirement 6.2.4 where the application participates in a payment flow.

This letter reflects the security posture of the assessed environment at the conclusion of the retest window. It does not constitute a guarantee against future compromise, and it does not extend to changes made to the environment after that date.

Yours faithfully,

[Name], Practice Manager — Offensive Security
Cetonix · info@cetonix.com · Verification of this letter may be requested at the address above, quoting CTX-PT-2026-0437.

END OF REPORT