

SERVICE DATASHEET

Mobile Application Security

Manual mobile testing assessed against the OWASP Mobile Application Security Verification Standard: what the binary gives away, what the app leaves on the device, what it does under instrumentation, and whether the back end behind it trusts the client.

BEST FOR	Consumer and enterprise apps; pre-release and pre-certification review; fintech and health apps handling regulated data
YOU RECEIVE	MASVS-mapped findings, runtime proof, prioritised fixes, retest letter
COVERS	iOS, Android, OWASP MASVS

The client is in the attacker's hands, not yours

Every other target you own. A mobile app you give away: the bundle ships to anyone with the store listing, and from that moment the binary, the local data store and the runtime are an attacker's to take apart at leisure. We start where they do — unpack the shipped artefact and read it — because what is compiled into it cannot be revoked by a server-side change. On one engagement that was enough to recover backend credentials granting a live, authenticated session to an internal customer-data API; the full chain is published.

The second half of the work is the part scanners never reach: hooking the running app to see what its defences actually hold up to. Root and jailbreak detection, certificate pinning and anti-tampering are controls we expect to be able to defeat; the question we answer is what each one is protecting and how much is left when it is gone. We test the app and the API it talks to as one system, because a client-side control is only a control if the server does not take the client's word for it.

What we test

- The shipped binary: secrets, keys, endpoints and logic recovered by reverse engineering
- Data at rest: local databases, keychain and keystore use, caches, backups and logs
- Cryptography and key derivation, including keys derivable from the device itself
- Transport: TLS validation, certificate pinning and the configuration that disables it
- Platform surface: deep links, IPC, exported components, WebViews and embedded local servers
- Resilience: root and jailbreak detection, anti-debugging, anti-tampering and runtime hooking
- Authentication, session handling and biometric gating
- The back-end API the app talks to, tested as part of the same system

How it runs

- **Scope and builds** — Platforms, build artefacts, test accounts and device or emulator access agreed up front.
- **Static analysis** — We unpack and reverse the shipped bundle: the secrets, endpoints, key material and logic it gives away.
- **Dynamic and runtime** — Instrumented testing on device — hooking, interception, storage inspection and control bypass — followed through into the API behind the app.
- **Report and retest** — Findings mapped to the MASVS control groups, prioritised, then verified and confirmed in writing.

What you get

- **Executive summary** — Business-level risk, written for leadership and boards.
- **MASVS-mapped report** — Every finding with its severity, evidence, reproduction steps and the MASVS control group it sits in — plus the recovered artefacts and intercepted traffic behind it.
- **Prioritised fixes** — Developer-ready remediation for the platform and framework you actually build on.
- **Retest letter** — We verify your fixes and confirm closure in writing.

Methodologies and references we draw on

OWASP MASVS (Mobile Application Security Verification Standard) · OWASP MASTG (Mobile Application Security Testing Guide) · OWASP Mobile Top 10 · OWASP API Security Top 10 · NIST SP 800-163 (vetting mobile applications) · CWE · CVSS and EPSS for severity context

Senior testers. Evidence for every finding. Retest included. Minimum five testing days. Scope and sample reports — pxlsec.com/services/mobile-application-security.