What is mobile app penetration testing?

Mobile app penetration testing is a controlled attack on an iOS or Android application and the backend API behind it, to find what an attacker could actually do. It is different from web application testing in one structural way: the attacker holds the client. They install the app on a device they fully control, decompile it, observe and modify it at runtime, and bypass anything the app tries to enforce locally. A mobile pentest covers three layers: the app binary, the data it stores and transmits, and the backend API.

HOW IT WORKS

01 Why is mobile testing different from web testing?

One structural difference changes everything: in mobile, the attacker holds the client.

In a web application, the server controls the page and the attacker only sees what the server sends. In a mobile application, the attacker installs the app on a device they fully control. From there they can:

- Decompile the binary and read the code, embedded secrets, API keys, and logic.
- Run the app on a rooted or jailbroken device where every security boundary is theirs to remove.
- Hook into the running app with instrumentation tooling and change its behaviour live: skip a check, change a value, dump memory. See Frida.
- Intercept the app's network traffic even when the app tries to prevent it. See certificate pinning bypass.

The practical consequence: any security control the app tries to enforce on the device is a speed bump, not a wall. The real security boundary is the backend. A mobile pentest that only checks the app and ignores the API misses where the damage is.

02 What does a mobile pentest find?

The recurring high-impact findings from real engagements:

SOURCES

[1] OWASP MASVS

[2] OWASP MASTG

[3] OWASP Mobile Top 10

[4] NIST SP 800-163 Vetting the Security of Mobile Applications

- Hardcoded secrets in the binary. API keys, credentials, encryption keys, and third-party tokens embedded in the app and trivially extracted by decompiling it.
- Insecure local storage. Sensitive data (tokens, personal information, cached responses) stored unencrypted on the device.
- Weak or bypassable transport security. Traffic that can be intercepted because pinning is absent or bypassable.
- Backend authorisation flaws. The API trusts the app to enforce rules the attacker can ignore. This is where the BOLA / IDOR class shows up in mobile.
- Client-side trust. The app enforces a business rule (a price, a limit, a permission) on the device, and the backend does not re-check it.
- Bypassable protections. Root / jailbreak detection, certificate pinning, and anti-tamper that the tester defeats to demonstrate the protection is not a real boundary.

03 What standards is a mobile pentest measured against?

Two OWASP standards define the bar:

- OWASP MASVS (Mobile Application Security Verification Standard). The checklist of what a secure mobile app should do, organised into categories like storage, cryptography, authentication, network, and resilience.
- OWASP MASTG (Mobile Application Security Testing Guide). The how-to manual that shows the techniques for verifying each MASVS requirement.

A good engagement maps findings to MASVS so the customer can see exactly which requirements pass, which fail, and what to fix. Compliance frameworks (PCI MPoC, app-store requirements, enterprise procurement) increasingly reference MASVS directly.

04 How does SecureLayer7 run a mobile pentest?

Four phases.

Phase 1, static analysis. Decompile the binary, read the code, hunt for embedded secrets, map the data the app stores, review the protections it ships with.

Phase 2, dynamic analysis. Run the app on an instrumented device. Observe and modify it at runtime. Test local storage, inter-process communication, deep links, and the protections (pinning, root detection) by attempting to bypass each.

Phase 3, network and API. Intercept the app's traffic, map every endpoint, and run a full API penetration test against the backend: authentication, authorisation (BOLA), input validation, rate limiting.

Phase 4, reporting. Findings mapped to OWASP MASVS, each with reproducible evidence and the specific code, configuration, or architectural change required, plus a free re-test of fixes.

The deliverable names which MASVS requirements pass and fail, so the customer and any auditor can see the coverage.

Test your iOS and Android apps end to end.
securelayer7.net/learn/mobile-security/what-is-mobile-app-pentesting

Open online