What is Frida?

Frida is a free, open-source dynamic instrumentation toolkit. It lets a tester inject code into a running application and change its behaviour live: read and modify memory, intercept function calls, replace return values, and trace what the app does. It works on Android, iOS, and other platforms, and it is the backbone of dynamic mobile-app testing. Because it operates on a running app on a device the tester controls, it is also the tool used to bypass on-device protections like certificate pinning and root detection.

HOW IT WORKS

01 Why does Frida matter for mobile security?

Frida matters because it makes concrete the central fact of mobile security: the attacker holds the client.

Any check the app performs on the device, Frida can observe and change. A check that decides 'is this device rooted?' can be hooked so it always returns 'no'. A check that decides 'is the certificate valid?' can be hooked to always say 'yes'. A value the app computes locally can be overwritten before it is used.

This is why on-device protections are speed bumps, not walls. Frida demonstrates, in a reproducible way, exactly how an attacker would defeat each protection. When a pentest report says 'root detection was bypassed', the bypass was almost certainly done with Frida or a similar tool, and the report includes the script so the finding is reproducible.

02 What do testers actually do with Frida?

The recurring uses on a real engagement:

- Bypass on-device protections. Hook root/jailbreak detection, certificate pinning, and anti-tamper so testing can proceed. See certificate pinning bypass and root/jailbreak detection bypass.
- Extract secrets from memory. Read encryption keys, tokens, and decrypted data the app holds in memory at runtime, even when it is not written to disk.

SOURCES

[1] Frida (open-source dynamic instrumentation toolkit)

[2] OWASP MASTG Dynamic Analysis

[3] OWASP MASVS Resilience

- Trace the app's behaviour. Watch which functions handle sensitive data and how, to understand the app's real logic faster than reading decompiled code.
- Manipulate client-side checks. Demonstrate that a business rule enforced on the device (a price, a limit, a permission) can be changed, proving the backend must re-check it.
- Test cryptography use. Observe how the app uses encryption keys and whether they are handled safely.

03 What about Objection?

Objection is an open-source toolkit built on top of Frida that packages many common mobile-testing tasks into ready-made commands. Where Frida is a general instrumentation engine that you script, Objection provides pre-built actions for the things testers do most often: bypassing pinning, bypassing root detection, dumping the keychain, inspecting storage.

For a tester, Objection is a convenience layer over Frida. The underlying mechanism is the same. A complex or app-specific bypass still needs a custom Frida script; the routine ones are faster through Objection.

04 What does this mean for developers?

If you are building a mobile app, the existence of Frida should shape your threat model in one specific way: assume any check your app performs on the device can be defeated.

Concrete implications:

- Do not rely on client-side enforcement for security decisions. If the app decides a user's permissions, a price, or a limit, the backend must independently re-check it. The client decision is a UX optimisation, not a security control.
- Treat on-device protections as defence in depth, not as boundaries. Root detection and pinning raise the cost of an attack and are worth having for high-value apps, but they do not stop a determined attacker with Frida.

- Keep secrets off the client where possible. Anything in the app's memory can be read at runtime. Minimise what the client holds.

The security boundary is the server. Frida is the proof.

See exactly how your app's protections hold up.
securelayer7.net/learn/mobile-security/frida-basics

Open online