Android vs iOS pentesting.

Android and iOS apps share the same backend but differ in almost everything on the device. Android apps are packaged as APK or AAB files, are usually easier to decompile, and run on a more open platform where rooting and instrumentation are straightforward. iOS apps are packaged as IPA files, are harder to decompile cleanly, and run on a more locked-down platform where jailbreaking is required for deep testing. A finding on one platform does not guarantee the same finding on the other, so an app shipped on both is tested on both.

HOW IT WORKS

01 What is specific to testing Android?

Android is the more open platform, which makes testing more straightforward.

- Decompilation is easy. Android apps compile to a bytecode that decompiles back to near-readable Java or Kotlin. Hardcoded secrets, logic, and API endpoints are usually quick to recover.
- Rooting is accessible. Test devices and emulators can be rooted readily, giving the tester full control.
- The component model is an attack surface. Android apps expose components (activities, services, broadcast receivers, content providers) that other apps can interact with. Exported components that should be private are a common finding.
- Inter-process communication via Intents. Deep links and Intents are a common path for an attacker app or a crafted link to reach functionality it should not.
- Storage defaults. Shared preferences and local databases are unencrypted by default. Sensitive data stored there is a frequent finding.

02 What is specific to testing iOS?

iOS is the more locked-down platform, which makes testing require more setup but does not make apps secure by default.

SOURCES

[1] OWASP MASTG Android Testing

[2] OWASP MASTG iOS Testing

[3] OWASP MASVS

- Decompilation is harder but not impossible. iOS binaries are compiled to native machine code, which is harder to read than Android bytecode, but the strings, embedded secrets, and structure are still recoverable.
- Jailbreaking is required for deep testing. A jailbroken device or a specialised testing setup is needed to instrument the app and inspect the file system.
- The Keychain is the right place for secrets. iOS provides the Keychain for secure storage. A common finding is sensitive data stored outside the Keychain, in plist files, in the app's documents directory, or in caches.
- URL schemes and universal links. The iOS equivalent of Android's deep-link attack surface. Crafted links can reach app functionality.
- Pasteboard and backgrounding leaks. iOS-specific data-exposure findings, like sensitive data left on the shared pasteboard or captured in the app-switcher snapshot.

03 Does the tooling differ?

Some tools are cross-platform, some are platform-specific.

- Cross-platform. The open-source instrumentation toolkit **Frida** works on both, and is the backbone of runtime testing for either platform. Network interception tooling is also shared.
- Android-specific. Decompilers and bytecode tools, the Android Debug Bridge, and emulator-based testing.
- iOS-specific. Jailbreak tooling, the iOS file-system access tools, and binary-analysis tools tuned for native code.

The technique is the same on both platforms (static analysis, dynamic analysis, network and API testing); the specific tools and the effort to set up the test environment differ.

04 If we can only test one platform, which?

Test the platform with the larger user base or the more sensitive use case first, but understand the limit: testing one platform only covers that platform's client. The backend API is shared

and gets tested either way, which is the most important part. The platform-specific client findings (storage, deep links, protections) only apply to the platform you tested.

For most teams the right answer is to test both clients (because the platform-specific findings differ) while testing the shared API once. The incremental cost of the second platform is lower than the first because the API work is already done.

Test Android, iOS, and the shared backend.
securelayer7.net/learn/mobile-security/android-vs-ios-pentest

Open online