1. Setting Up the Development Environment

Goal: Install and configure all tools needed for Android malware analysis and development (educational purposes only).

Required Tools:

- **Android Studio – Full-featured IDE with emulator and Gradle integration**
- **Android SDK + AVD – For creating and running Android Virtual Devices**
- **Genymotion – Alternative emulator with root access support**
- **Frida – Dynamic instrumentation tool**
- **ADB (Android Debug Bridge) – Command-line interface to interact with devices**
- **Optional: apktool, Jadx, MobSF, dex2jar for static analysis**

2. Preparing the Victim Environment

Goal: Build a secure, isolated test environment for running and analyzing malicious APKs.

Option A: Using Android Studio AVD

1. **Open AVD Manager from Android Studio.**
2. **Create a new device (e.g., Pixel 4).**
3. **Use a system image with `x86_64` and Google APIs (Android 9+ recommended).**
4. **Configure proxy to redirect traffic through Burp Suite.**
5. **Launch emulator with `-writable-system` flag if needed.**

Option B: Genymotion VM

1. **Create a new virtual device with root access.**
2. **Enable ADB bridge to communicate with host tools.**

3. Set proxy to route traffic to Burp Suite or mitmproxy.
4. Install custom CA certificate if intercepting HTTPS.

Option C: Physical Device (Optional)

- Enable Developer Options and USB Debugging.
- Root the device using tools like Magisk.
- Connect via `adb` and use for realistic testing.

3. Creating and Configuring a Basic APK

Goal: Build a simple educational APK from scratch for controlled testing.

Steps:

1. Start a new project in Android Studio.
2. Edit `AndroidManifest.xml` to declare basic permissions and components.
3. Add a simple Activity that mimics malicious behavior (e.g., logging data).
4. Build the APK in debug mode.
5. Sign the APK manually or with debug keystore.

4. APK Anatomy: AndroidManifest.xml and Smali

Topics Covered:

- Internal APK structure: `classes.dex`, `res/`, `AndroidManifest.xml`
- Understanding the manifest: permissions, components, intents
- Introduction to Smali (Dalvik bytecode format)
- Editing Smali for reverse engineering, behavior injection, or bypassing logic
- Practical Toolchain Overview

5. Dangerous Permissions and Access Abuse

Key Concepts:

- Common dangerous permissions (`READ_SMS`, `RECORD_AUDIO`, `ACCESS_FINE_LOCATION`)
- How malware abuses permissions:
  - Tricking users into granting access
  - Exploiting `AccessibilityService`
  - Silent data exfiltration or sensor activation

6. Covert Access to Camera and Microphone

Implementation:

- Using `Camera2` or `MediaRecorder` APIs to record secretly
- Running capture code in background services
- Avoiding system warnings or user prompts

7. Screen Capture and Overlay Attacks

Techniques:

- Using `SYSTEM_ALERT_WINDOW` for fake overlay UIs (e.g., fake login screens)
- Capturing screen content via `MediaProjection API`
- Example scenarios: phishing overlays, invisible UI hijacking