All this text wasn’t written by me (Tiago a.k.a TigaxMT).

All of it was transcribed by me from HackerOne video lessons.

All credits and thanks go to them.

Android Quickstart

STRUCTURE OF AN ANDROID APP

APK CONTAINER

Android apps come in the APK format, which is really a ZIP file. Inside it you'll find the signature, the application code, and all it resources. This can be unzipped with any standard archive tool, if you just rename to .zip. However, apktool will automatically unpack this more deeply.

DEX

Dex files contain compiled Java/Kotlin code for an Android application. In the case of standard native Android apps, will be all the code. This can disassembled with apktool and decompiled with likes of dex2jar.

MANIFEST

The AndroidManifest.xml file contains key information regarding the name and version of the app, its requested permissions, activities, intents, and more. Much of this information is useful during application testing, and apktool decodes this to a readable format.

RESOURCES

An Android app contains many resources, from images and strings, to XML files which describe UI layouts, These are worth searching through, if only to find unused layouts(e.g debug interfaces) and strings which may point to bugs elsewhere.

TOOLS

ANDROID STUDIO

https://developer.android.com/studio

This is standard Android development environment. Within it, you can both build applications and run them in an emulator. The ability to build small test applications is very useful for mobile testing.

ANDROID EMULATOR

Within Android Studio it's possible to create virtual Android devices, which run via their emulator. Many options are given here, but the most important is likely the choice between x86(Intel architecture) or ARM.

X86 will almost always run faster and more smoothly, as it's not actually emulated. Rather, it's run as a simple virtualized machine, allowing high performance.
However, some phone-specific applications won't run on x86 due to their native code libraries being built for ARM.

GENYMOTION

https://www.genymotion.com/

This is an alternative to the typical Android emulator. It is faster and nicer than the standard emulator, but it is primarily a commercial product. A free version is available, however this is very much optional.

APKTOOL

https://ibotpeaches.github.io/Apktool/

This lets you unpack and decode APK files to a usable form, which can then be manipulated and rebuilt into a working application. This is useful for inserting debugging code, inspecting resources, and more.

DEX2JAR

https://github.com/pxb1988/dex2jar

This tool will translate an APK or DEX file to a JAR, which can then decompile.

JD-GUI

http://jd.benow.ca/

This is a great compiler for Java .class and .jar files, which makes it optimal for turning an APK into readable Java code after running dex2jar.

FRIDA

https://www.frida.re/

This lets you instrument applications, meaning you can hook and inspect functions that are called. This can be used for everything from watching files that are opened or monitoring network connections, to disabling certificate pinning.

BURP + CA CERTIFICATE SETUP

PROXY SETTINGS

For the Android emulator, proxy settings are located on the settings section of the “Extended controls” screen. These can be set to localhost and the port Burp is listening on.
No further configuration should be required for the proxy itself.

For physical devices, proxy settings are under WiFi settings. Long-press the network you’re connected to, click “Modify network”, then the proxy settings will be on that dialogue.
Make sure that your proxy listener in Burp is set to either listen on all interfaces, or on the local network interface – not localhost only.

INSTALLING CA CERTIFICATE

For your device to trust Burp for SSL connections, you need to install the CA cert.
Once the proxy is enabled, go to http://burp/ in the browser and download the certificate from the top-right.
Then go to Android Settings, Security & location, Encryption & credentials, Install from SD card.

ROOTING

WARNING

Never root your primary device(s). They are at a seriously increased risk of compromise and your IT people at work will hate you.

Only root devices that you would willingly unlock and hand over to a stranger.

THE PROCESS

The rooting process is device-specific, ranging from a few simple commands to multiple exploits. If you have a choice of test device, go for a Google phone/tablet for ease of rooting.

In general, searching for “<device model> root instructions” will get you the information you need.

DECOMPILATION

THE PROCESS

To convert the apk into a format readable by a Java decompiler, run: dex2jar -f path/to/app.apk

From here, you have a standard Java Jar file. Open resulting jar file (app-dex2jar.jar) in JD-GUI and you’ll see the Java Code.

RECOMMENDATIONS

JD-GUI is a good decompiler with a very limited interface. Once you load you Jar, use the Save All Sources option in the File menu to save all the decompiled .java files to disk. This will let you open the entire thing in a better editor of your choice. In many cases, you can turn the entire thing into a new Android Studio project.

TESTING TIPS

LOGCAT

When you install Android Studio and the Android SDK, you’ll receive a tool called adb. This is the official way to debug Android-based devices, both physical and virtual. The adb logcat command will display the running log of everything occurring on the device, often letting you see debug messages in flight.

DISABLE CERT PINNING

If you target application fails to connect after enabling your proxy, certificate pinning may be the problem.

There’s a great guide on different approaches to cert pinning removal, referenced on the Hacker101 website.

INTENT FILTER CHECK

Android’s Intent filters allow for an app to handle specific URL patterns or protocol schemas. These can lead to vulnerabilities where a website can open malicious payloads in the target application.
Example report: https://hackerone.com/reports/283063