Android App Reverse Engineering

When you want to make changes to an Android application, there are several steps that need to be taken. In this quick tutorial/overview we go over these steps, which tools you can use during these steps and what you need to pay attention for. This guide is by far not fully complete, so there might still be some research that needs to be done by yourself.

Disclaimer: We do not condone the use of this guide for illegal activities! When making important changes to Android applications always consult the developers of the application for permission to do so.

Step 1: Do I have the APK?

You do?!

- Cool! You can already move to the next step. Move along hackerman!

You don't?!

- Well you are most likely trying to do some bad things to an application on the play store then. For this you can use an application like Apk Extractor. Not sure whether you can use Apk Extractor to get the apk of Apk Extractor tho.. Food for thought.

Step 2: What do I do with the APK?

Well this entirely depends on what you wanna achieve.

I simply want to take a look at the code resources

- There is this nice nifty online tool that uses *jadx* to decompile the app for you: http://www.javadecompilers.com/apk
- If you rather do it yourself, you can download *jadx* and run some commands (it even has a nice GUI): https://github.com/skylot/jadx
- The tutorial ends here for you :) Have fun!

I want to change the code!

- Oh boy, you're in for a ride! There are several tools you'll need, so let's list them here. There are two ways to go about this, the GUI way or the command line way. If you choose GUI, I recommend downloading the optional tools as well.
    - **Apkstudio (optional):** This is my go-to pick for anything related to decompilation and recompilation with Android. It's an open-source IDE that combines all the tools we mentioned together into one big platform/UI to make it easier for you.
        - **Warning**: All the different tools still need to be downloaded separately, however *Apkstudio* will give you the required links. Having the jars of the other tools will suffice.

- **Apktool**: This tool allows you to both decompile and recompile android APK's. It's the main component for being able to make changes to an Android app.
- **Uber-signer**: This tool is used to sign your applications with a keystore (see later why required). No it has nothing to do with cars..
- **jadx** **(optional)**: Normally you shouldn't need *jadx*, but it is still recommended when using *Apkstudio* since it uses both *jadx* and *apktool* for decompilation.

Step 3: Decompiling, changing and Recompiling

Yeah yeah, I already mentioned decompiling the APK in the previous step. You already had to download so many tools, so let me consider this separated okay?!

How to look at Android code:

- **Smali code**: assembly code for the Dalvik VM, if you make changes to the app you will make it in this code.
- **Java code**: the java code used to build the apk. This java code could have been minimized/obfuscated using plugins like ProGuard, but it remains quite readable overall.

How to decompile, change, recompile:

We assume in these steps that you use the *Apkstudio*, the alternative command line tools will be mentioned in the subpoint.

1. Open the apk and select only smali and Java code (if you select resources, recompiling might fail due to incomplete resource extraction)
    a. This will create a directory with all the decompiled files.
    b. If you need the resources specifically, either decompile separately or take a further look using *jadx*.
    c. Command: ***apktool d -r -o "app-decompiled" app.apk***

2. Inspect the Java code to get a gist of how the application works. When you understand what needs to be changed by looking at the Java code, switch over to the smali version. For every Java file there will be a corresponding smali file that contains the assembly version of the Java code.

3. Change the smali code to get the behavior you desire, this can either be done in *Apkstudio* itself or using your favorite text editor and simply opening the corresponding file.

4. Recompile the folder back to an apk by selecting the "Project -> Build" option in *Apkstudio.*
    a. The apk will be in the dist folder of the project.

    b. Command: ***apktool b "app-decompiled"***

Step 4: Signing the APK

Yes.. The recompilation step is not the final part of the process. You could try and get the apk you have right now to run on your phone or an emulator, but 5 chances out of 7 this will result in failure.

The reason why is because Android has a mechanism for checking whether somebody is trying to reinstall an app update on your phone that is malicious. We call this the Android keystore, which is also used for signing apk's for deployment on the playstore.

To ensure your apk is deemed "safe" to install, you need to do the following steps:

1. Uninstall the old application from your device.
2. Sign your new apk using a keystore (*Apkstudio* "Project->Sign/Export"). If you leave the fields empty in *Apkstudio*, it will simply generate a debug keystore which will work fine.
    a. If you want to make one with specific parameters, you can either generate one using *keytool*, *Android Studio* or any other tool.
    b. Command: **java -jar uber-apk-signer.jar -a app-release.apk**
3. Put the apk on your device using any means necessary (just make sure that when you use things like google drive, your phone trusts apks downloaded through drive).

If all has gone well, you should have a modified application now! If not.. Well you either did something wrong or this tutorial has become obsolete. Regardless, the best of luck to you!