Major Steps that encounter while Reverse Engineering of an apk

Multidex Problem : So one of the major problem that we face while doing reverse engineering

- There are few methods available which are used to check the total no. of methods present in the targeted apk. So following is one of them
    - First change the apk file into dex file, and install dx application on your system or use the default dx application that comes with android studio sdk, -> build tools folder.

- Try to put minimum no. of sdks in you application as it can cause to increase the no. of methods in your application which is finally going to be merged with the Targeted app.
- Use methodcoudnt.com website to calculate the total no. of methods in your dependency from your gradle file
- Take out every jar file from your libs folder which you have added in your application and us dx tool to get the total no. of methods in each and every jar file

Package Name Changing : It is also one of the major step as all you change in the apk and finally successfully integrated your code in targeted apk then it finally need to be published on some marketplace whitch already had a apk with same name of package, so if you can't able to change package name its very difficult to publish that app with that name.

- Get the package name from Manifest file.
- Search for the package name of the app in all files
- Replace it with some non literals keyword in the project
- For eg: (Replace com.originalpackage.game with com.yourpackagename) and com/originalpackage/game with com/yourpackagename in all files

Graphics Check : All the ui design must be changed before putting it up to any market place, as google and other market place will ban your app if they find anything similar to other app that is already present on the market place.

- Decompile your app and check in assets folder of the app for the graphics, fonts and sound part.
- Make sure that every graphic, fonts and sound file must be changed a little bit so that it can easily change its hash value other wise platform like google easily recognised the assets with their hash value. In it

- **Change SHA of graphics:** Change SHA of all files in asset folder or res folder(Can ignore .fnt or ttf font files)
    - Change the sha of the assets(All kind of files like .plist,.csb, .tmx, .ogg, .mp3, .png) completely or either change it one by one. Can ignore font files and any lib file if it is there
    - Change the sha of sound files (NCH, wavepad sound editor)
    - Change the sha of resources file
    - In Res folder edit .9.png(9 patch) images in android studio not in photoshop, otherwise app would get crash or will not compile
    - To check SHA of all files use HashChecker.java given below:

- **Check In-App purchase:**To check in-app purchase is there in app or not
    - First check on playstore for in-app purchase
    - Check also is there a online game play in App
    - If IAP is available then search for In app purchase key by finding “MIIB” and “QAB” String
    - Replace IAP Key with new key
    - Find SKU Ids of products and either change them or make same on play publish store

- **Change Google analytics & Leaderboard:**
    - Find google analytics id by “UA-”
    - For leaderboard see “strings.xml” you will find a “app_id” and other “leaderboard_ids”. Change those with our Ids.

- **Remove cross promo ads:** Remove cross promotion ads in the app. It is totally depends on the game, approach can be:
    - First analyse all the places where cross promo ads are showing, by playing the game at least 7-8 levels
    - According to that start from Launcher Activity and try to find out classes or methods from which they may show Ads
    - If game is not proguarded then method names are like “showGameAd”, “showAd”, or you can guess like showWingame, showLoseDialog.
    - You can also put test logs in Smali to test the different places.
    - Every time u make some major changes compile and test before making more changes.

- **Change Their Ad IDs:** Change their all of the Ad Ids of all ad networks they have integrated
    - You can find Ids by finding adCompanies Initialize method calls. For that you have to see integration doc of respective Ad Company.
    - Once Id is found of one adNetwork find it in whole project and replace on all the places.

Useful commands in this process:

To decompile the app

./apktool d filename.apk

To Compile the app

./apktool b -f foldername

Jar signed command

jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-release-key.keystore my_application.apk alias_name

Zipalign command (Must before release)

zipalign -fv 4 oldapkname.apk newapkname.apk

Tips to Remember while reverse engineering of an app

- Never change the R file of the app that you are RE, Try to use every string from a constant file in you java class and make each and every layout dynamically with your java code.
- Put every drawable image file in asset folder of your sdk as it will help to reduce the chances to create extra id in your image.
- If any image file that need to used only from drawable as sometime like In Notification the small icon can only be set by providing int value no the path of the file, so try to look for the same kind of image that is already present in the targeted apk and use its id from R file and change it with your id of that Image.
- Merge your menifest file with targeted app manifest very carefully double check that all the permission that you are asking is not present already in the file, put your all activity, services, receiver very carefully.
- While changing try to test your apk after every small change as it will help us to know that where our app is actually crashing and where went something wrong.
- Use apktool to recompile the app and the app, and the compiled app with apktoll will found in “appfolder/dist” path.
- Use jarsigner to signed your apk so that it can be parsed by android and can install on a real device other wise it will always give you parse problem error.