Reverse Engineering Steps

- **Choose Game**:Search for the same kind of multiple games and download their APKs

- **Decompile Games and Check for Graphics**: Decompile downloaded APKS and check if graphics are available in png and jpg form in assets or res folder. If pngs are available then go to next step.

- **Try Changing Package Name:** Change package 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

- **Test after changing package name:** Compile and install game after changing package name check if game is working fine?

- **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.

OR

- **Integrate Our SDK:** Merging the game with our own code
  **Merge manifest file,
  - Permission if anyone added
  - Any activity that you have added in the app
  - Don't put our launcher activity in their manifest.
  - Any service that you have added or added by other adnetwork

  **Merging smali code
  - Whatever the new sdk that you have added in the code
  - And the code that you have added
  - **** Never copy your R file in the smalli of other code
  - Mostly the autogenerated files should not be added to the new one, It may cause to crash the app
  - Match you google play services with google play services that you are coding to show ad network
  - Merge the assest that you have used in the code, As assests cause minimum no. of conflicts in the smali code

  ** Initialize our SDK
  In onCreate method of launcher activity of game initialize our SDK and get the context from there. By it our code will be activated.

  ** Call our Show Ad Methods,
  Search for the game points where we have to show ads(same as you removed cross promo ads). From these points call our static Show ad methods.

  Finally Compile and check if all functionality is working fine.

- **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 asests(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:

Useful commands in this process:

To decomple 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
```

Class to compare SHA of assets and res

HashChecker.java

```
package com.hashchecker;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
import java.util.StringTokenizer;

import javax.xml.bind.annotation.adapters.HexBinaryAdapter;

public class HashChecker {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String path1, path2;

         Scanner in = new Scanner(System.in);

          System.out.println("Enter First path");
          path1 = in.nextLine();

          System.out.println("Enter Second path");
          path2 = in.nextLine();


          Map<String, String> map1=new HashMap<>();
        makeHashMap(new File(path1),map1);


           Map<String, String> map2=new HashMap<>();
        makeHashMap(new File(path2),map2);



            for (Map.Entry<String, String> entry : map1.entrySet())
            {
//                System.out.println("Map 1 entry key::"+entry.getKey());

              if(map2.containsKey(entry.getKey())){


              if(map2.get(entry.getKey()).equals(entry.getValue())){


if(entry.getKey().contains("common_signin")||entry.getKey().contains("common_signin")|
|entry.getKey().contains("ic_plusone")||entry.getKey().contains("facebook")||entry.getKe
y().contains("by_google")||entry.getKey().contains("common")){

                }
                else{
                    System.out.println(entry.getKey());
                }

              }
              }
//                System.out.println(entry.getKey() + "/" + entry.getValue());
            }

  }
  public static void makeHashMap(File node,Map<String, String> map){



    if(node.isDirectory()){
        String[] subNote = node.list();
        for(String filename : subNote){
            makeHashMap(new File(node, filename),map);
        }
    }
    else{
```

```
//           System.out.println(node.getAbsoluteFile());
            try {

                String filePath=node.getAbsolutePath().toString();
                String[] bits = filePath.split("/");
                String lastOne = bits[bits.length-1];
                String secondLast=bits[bits.length-2];
                map.put(secondLast+"/"+lastOne,calcSHA1(node) );
            } catch (NoSuchAlgorithmException | IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }

    }


    private static String calcSHA1(File file) throws FileNotFoundException,
   IOException, NoSuchAlgorithmException {

MessageDigest sha1 = MessageDigest.getInstance("SHA-1");
try (InputStream input = new FileInputStream(file)) {

   byte[] buffer = new byte[8192];
   int len = input.read(buffer);

   while (len != -1) {
       sha1.update(buffer, 0, len);
       len = input.read(buffer);
   }


//    System.out.println(new HexBinaryAdapter().marshal(sha1.digest()));

   return new HexBinaryAdapter().marshal(sha1.digest());
}
}

}
```