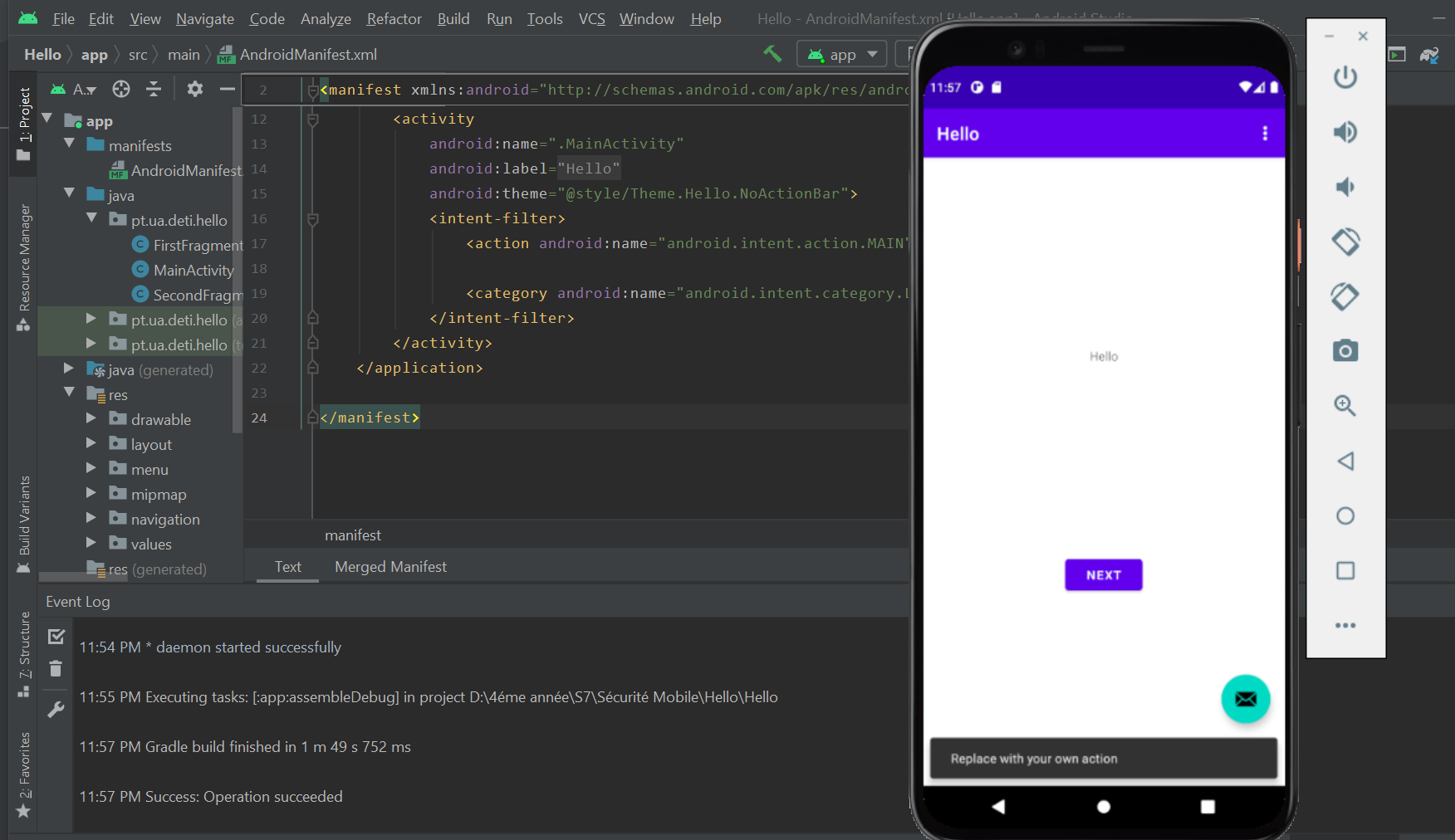
**Name:** Alaoui Belghiti Hanaa

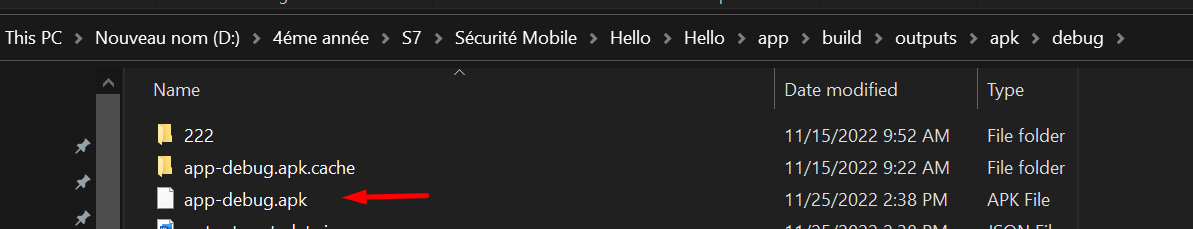
**Class:** SICS4

EXERCICE 1 :

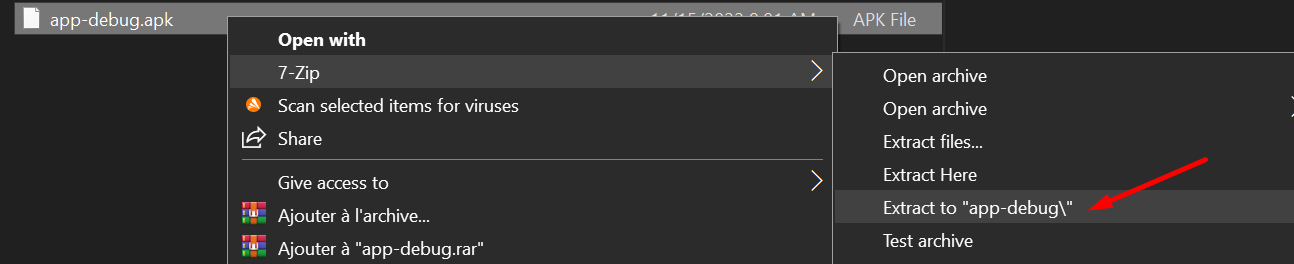
Launching the application in android studio:



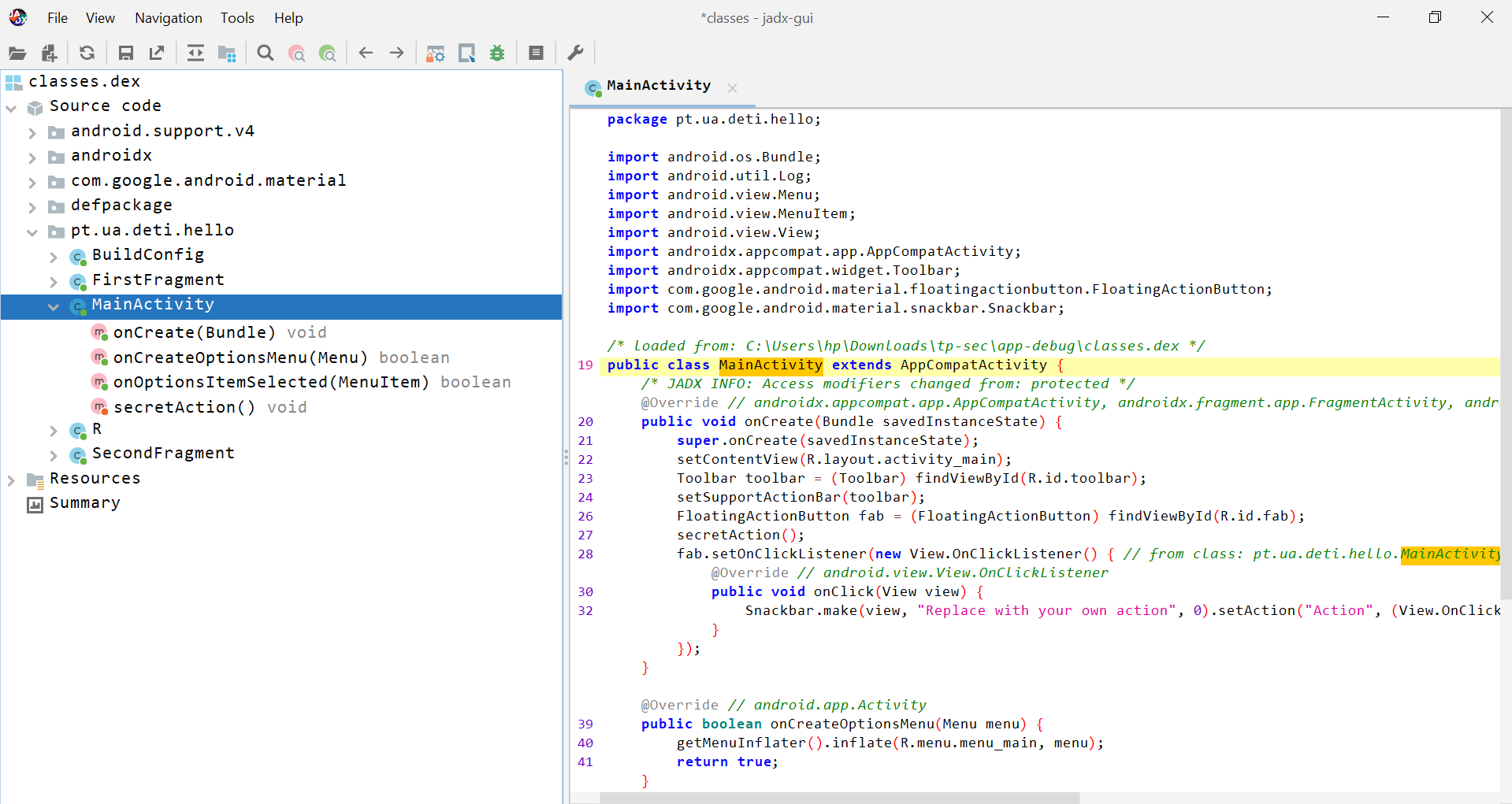
App-debug.apk’s location in the build folder:



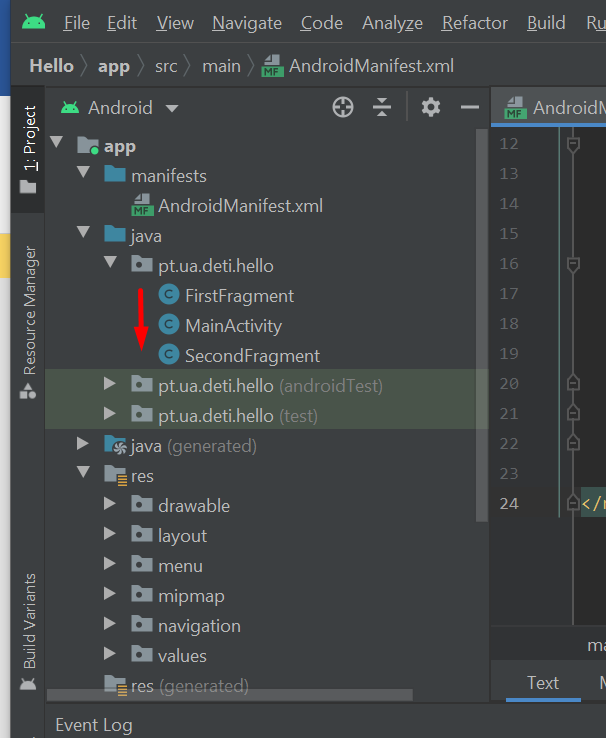
Decompressing the Apk with 7-Zip:



Opening the « classes.dex » file inside jadx application:

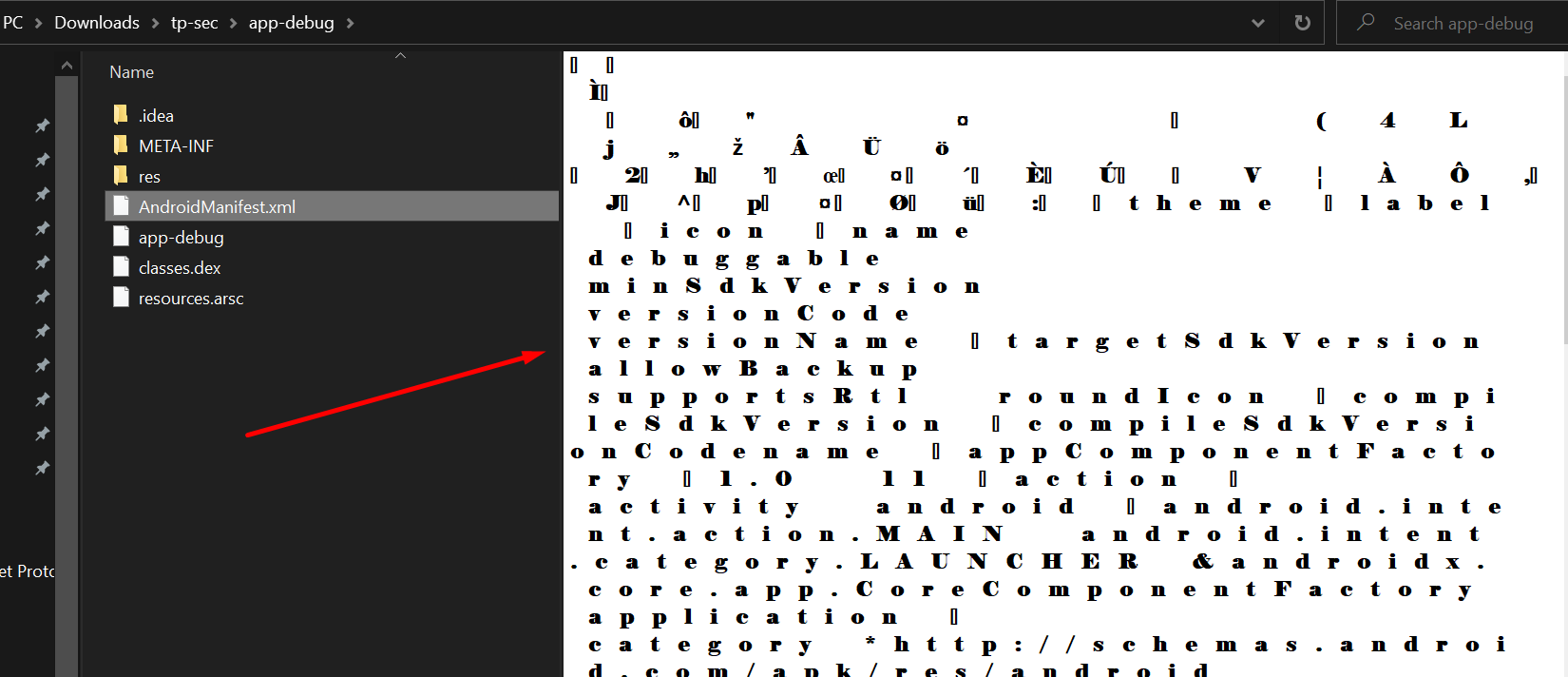


Comparing it to the original code that’s located inside android studio:



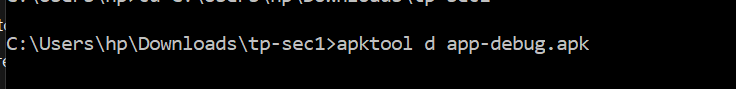
Noticing that there are different classes compared to classes.dex

Checking the “androidmanifest.xml” file’s contents after decompressing with 7-Zip:

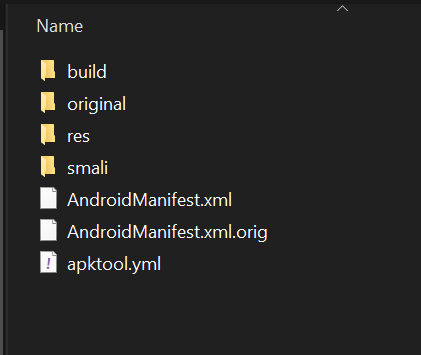


As we can see here, it is crypted.

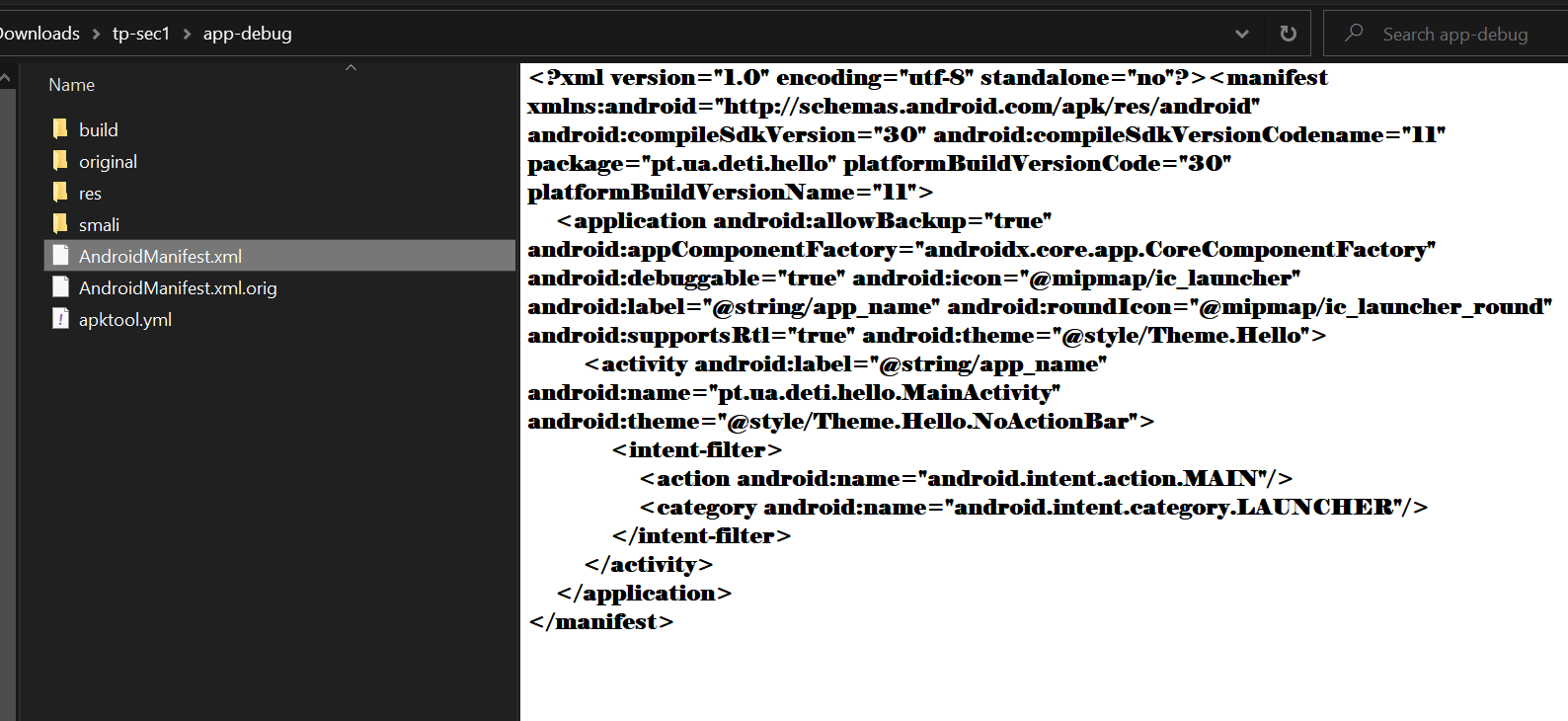
Decompressing the Apk with apktool instead:

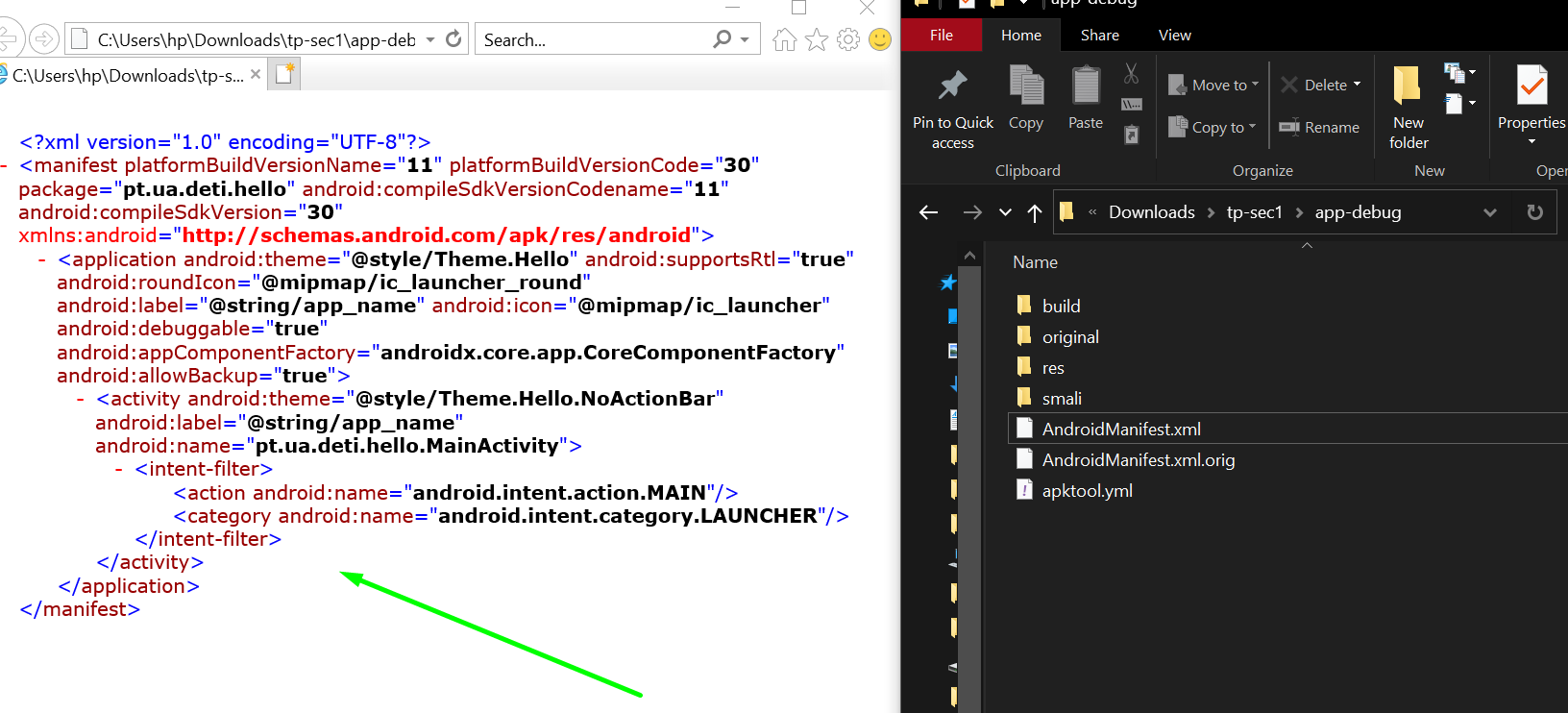


**Output :**

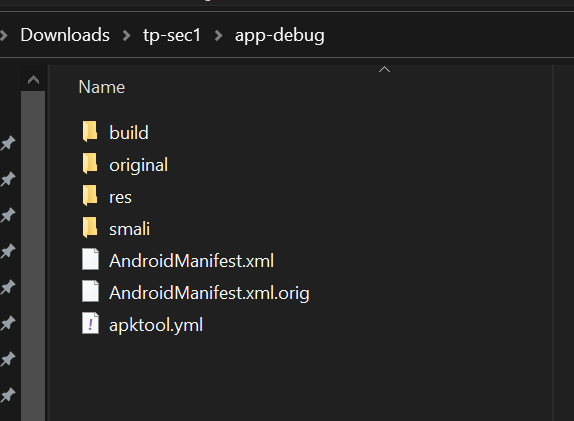


Now we will check the androidmanifest.xml file again:



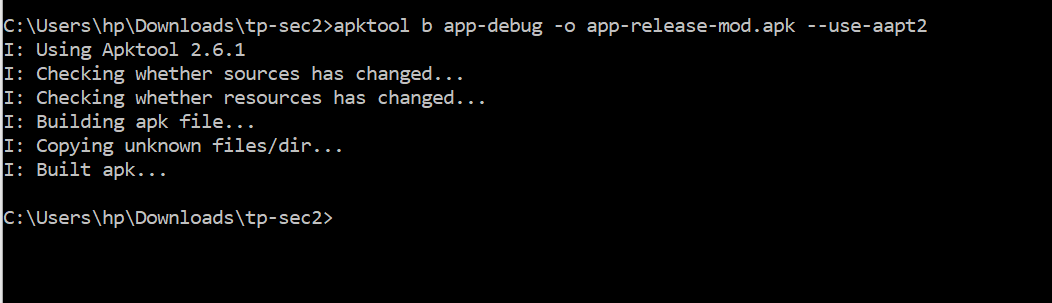


We notice that after decompressing it with Apktool, it is now readable.



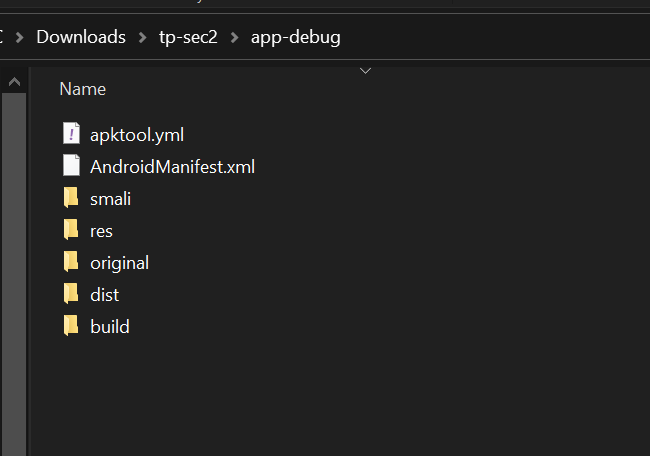
« smali » folder and other files have been added after decompressing with apktool, which shows the difference between it and the previous decompressing method.

After having moved the app-debug folder into a new ‘”tp-sec2”, I have recompiled it using the following command:

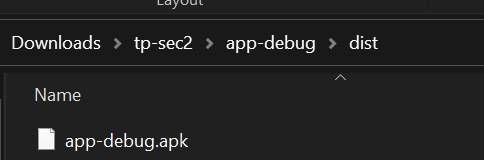


« built Apk » means that the app has been successfully created.

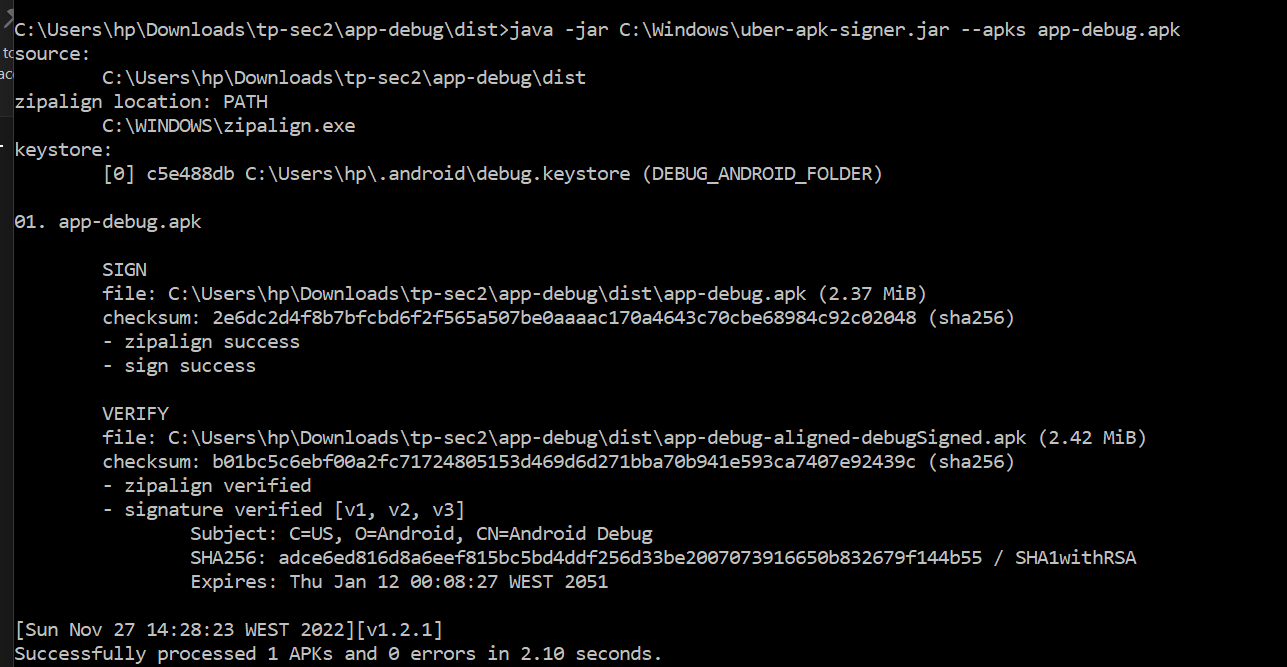
As we can see in the result below, a new folder inside the app-debug folder has been created, it is called” dist” and its where we will find the newly generated Apk.



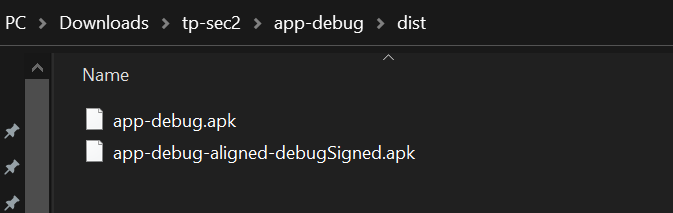
The Apk that has been created inside that folder:



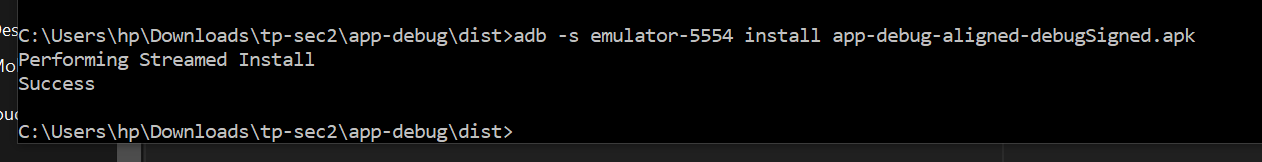
And now I will sign that Apk using the following command:



As you can see now, a new Apk has been created in the dist folder which is the signed Apk.

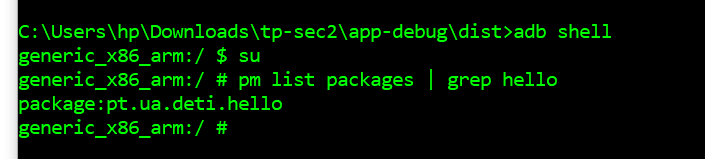


As for now, I will download the signed Apk into my emulator using the following command:

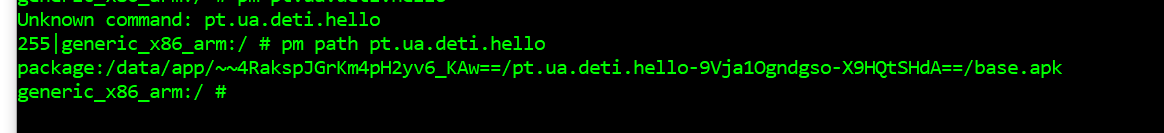


**EXERCICE 2 :**

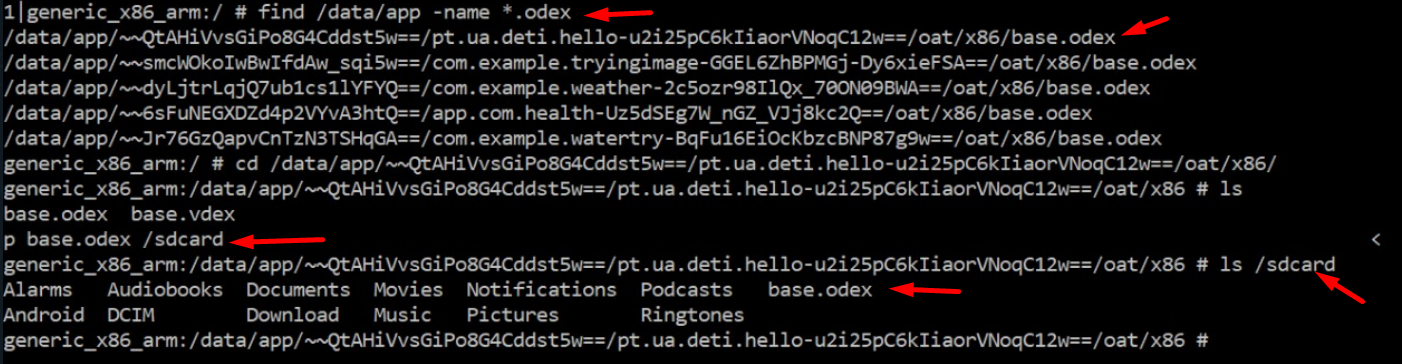
Navigating into the android shell via the adb shell command and listing the packages that belong to the hello app:



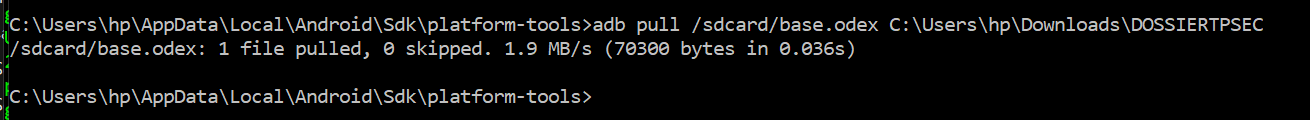
Looking for the base.odex file with the following command:



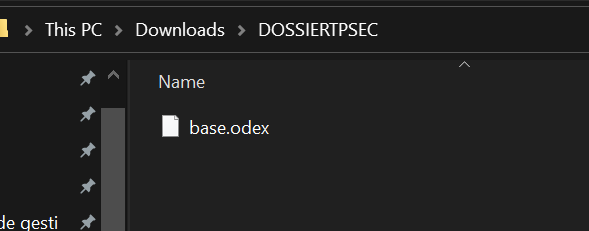
As You can see it is not found when we use that command, therefore, we will use a different approach in order to obtain it, that is, using the find command, as follows:



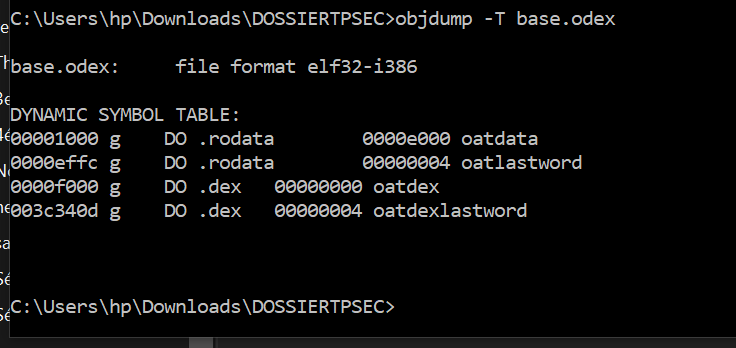
After finding the location, I’ve copied “base.odex” to the sdcard, then I’ve pulled it from the sdcard into a folder that I have on my computer, which is “ DOSSIERTEPSEC” as you can see here:



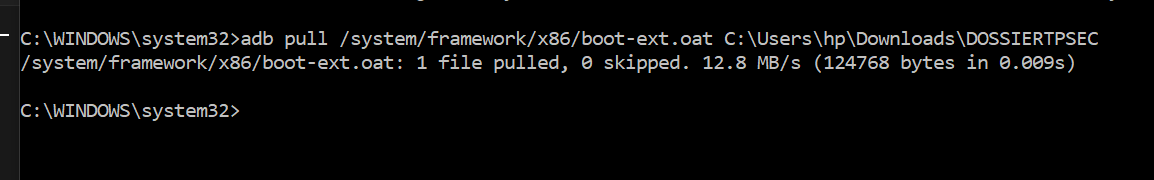
**This Is the result :**



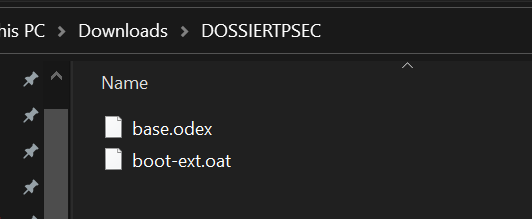
After obtaining it, I navigated to the base.odex location inside my folder and used the objdump command to show information regarding that file:



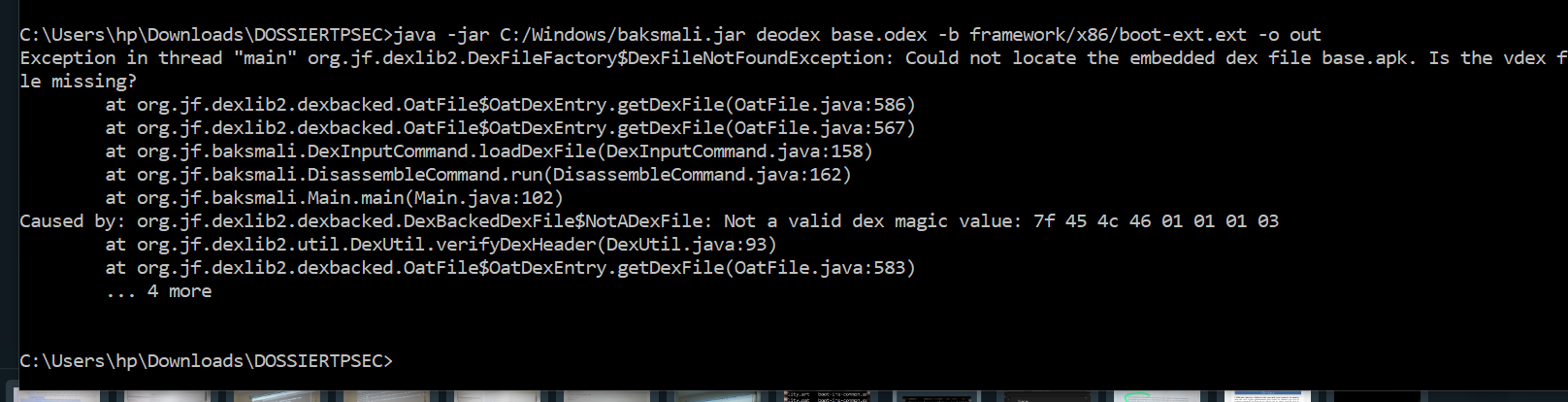
Then, I have looked for the boot.oat file, sadly I don’t have it, but I Have one that’s called boot-ext.oat, which is the one that I’m using for this example, I’ve pulled it into my “dossiertpsec” folder in order to preform operations on it as follows:



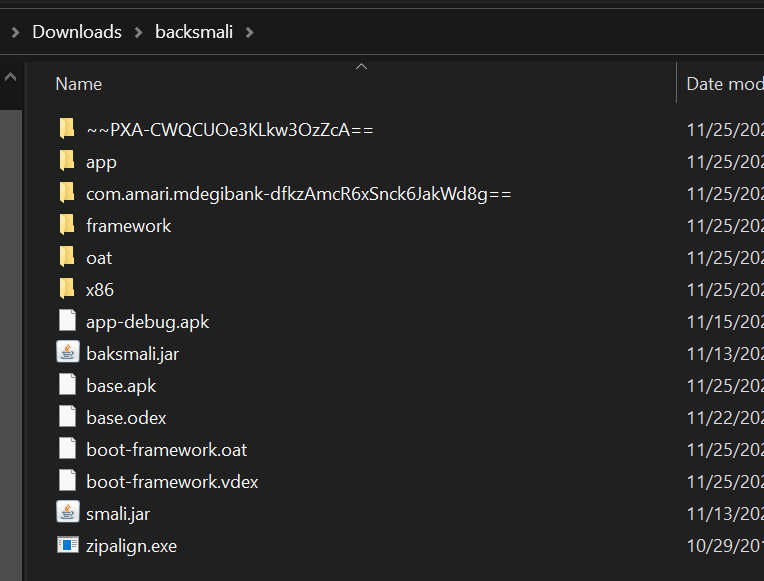
**Result :**



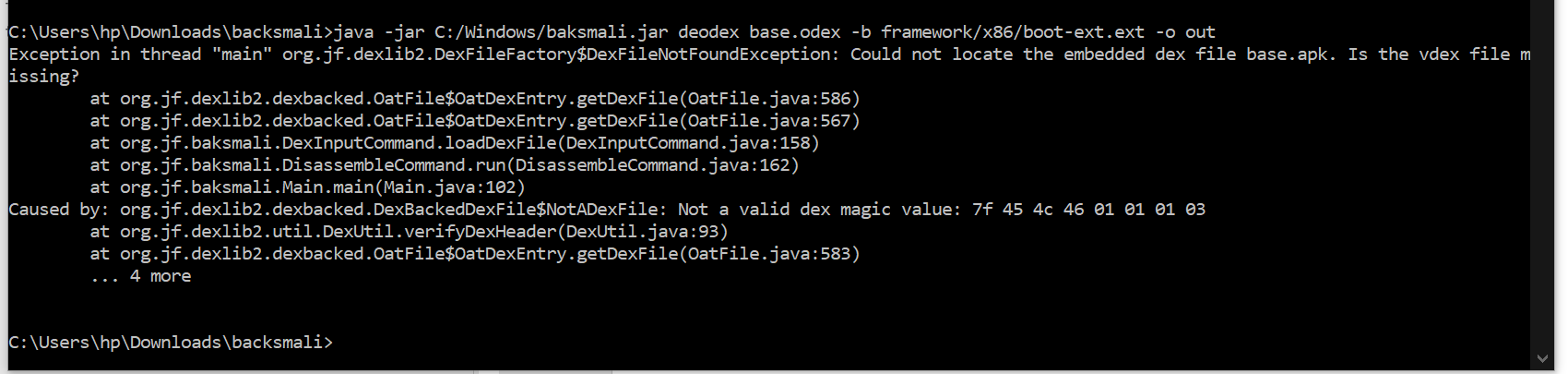
Now that I have obtained it, I’m going to use the baksmali.jar tool to read it, as you can see in the capture below, the command does not work because it says they couldn’t fetch the base.apk file



So, I’ve placed all the files inside a folder I named “backsmali”, including the “framework” folder and the x86 folder which I have pulled from my application, but the command still wouldn’t run, So I don’t know what to do.

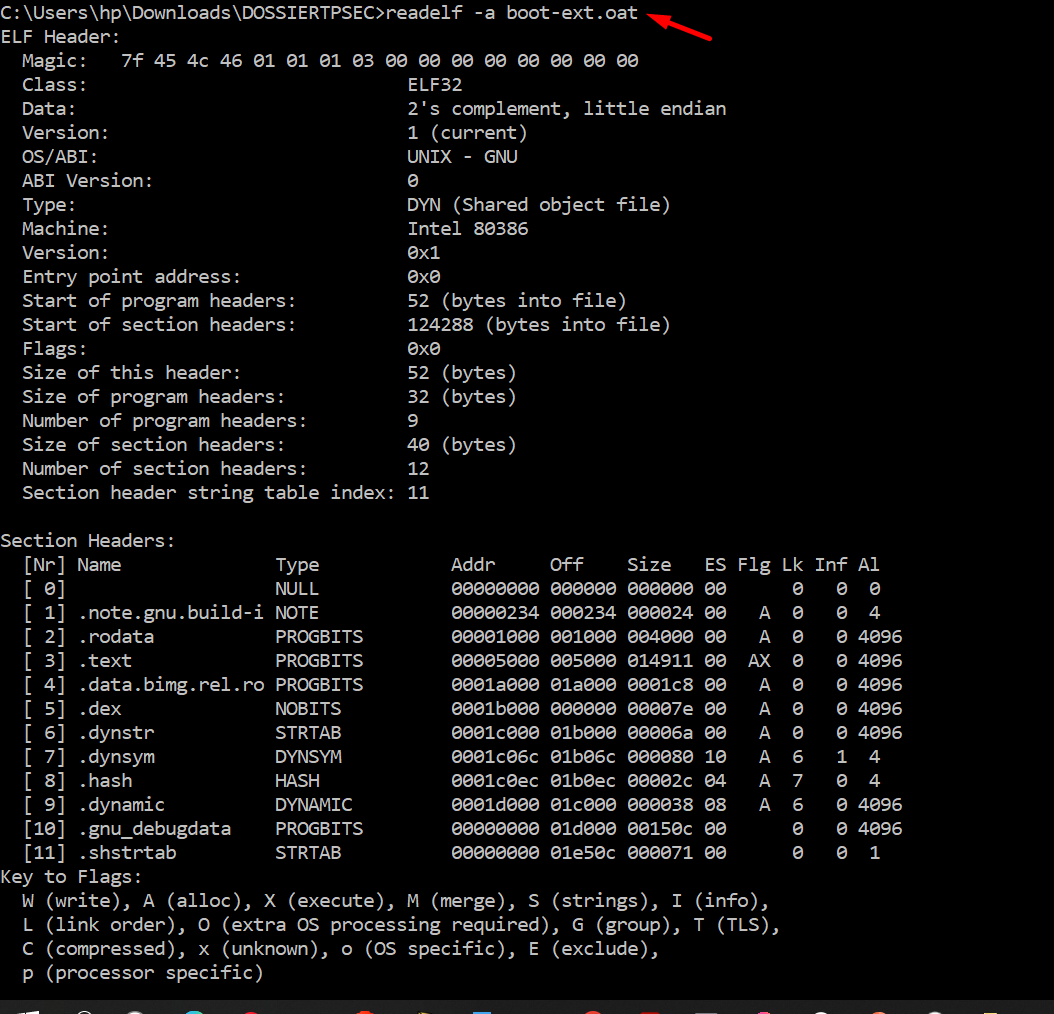


Still, nothing :

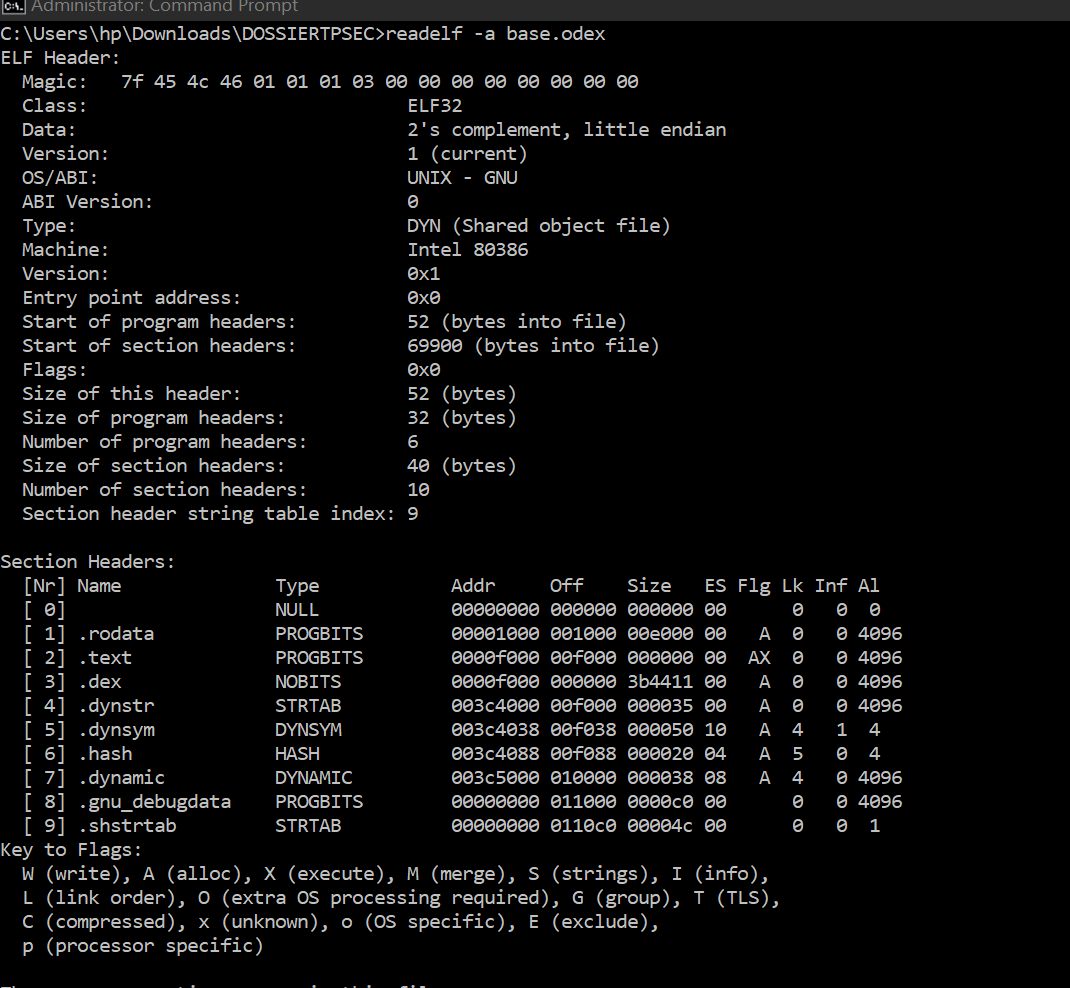


I’ve decided to use readelf tool instead and I read both base.odex and boot.oat files and got the following results:

**For boot-ext.oat:**



**For base.odex :**

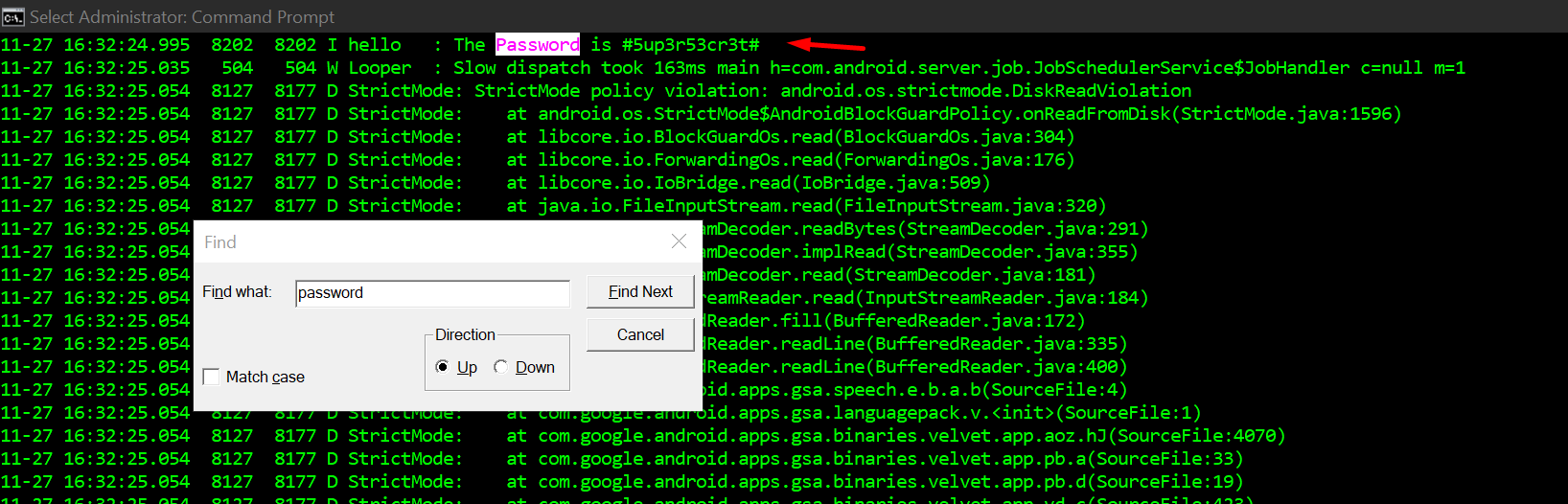


I have concluded that, both files have the same format.

**Exercise 3:**

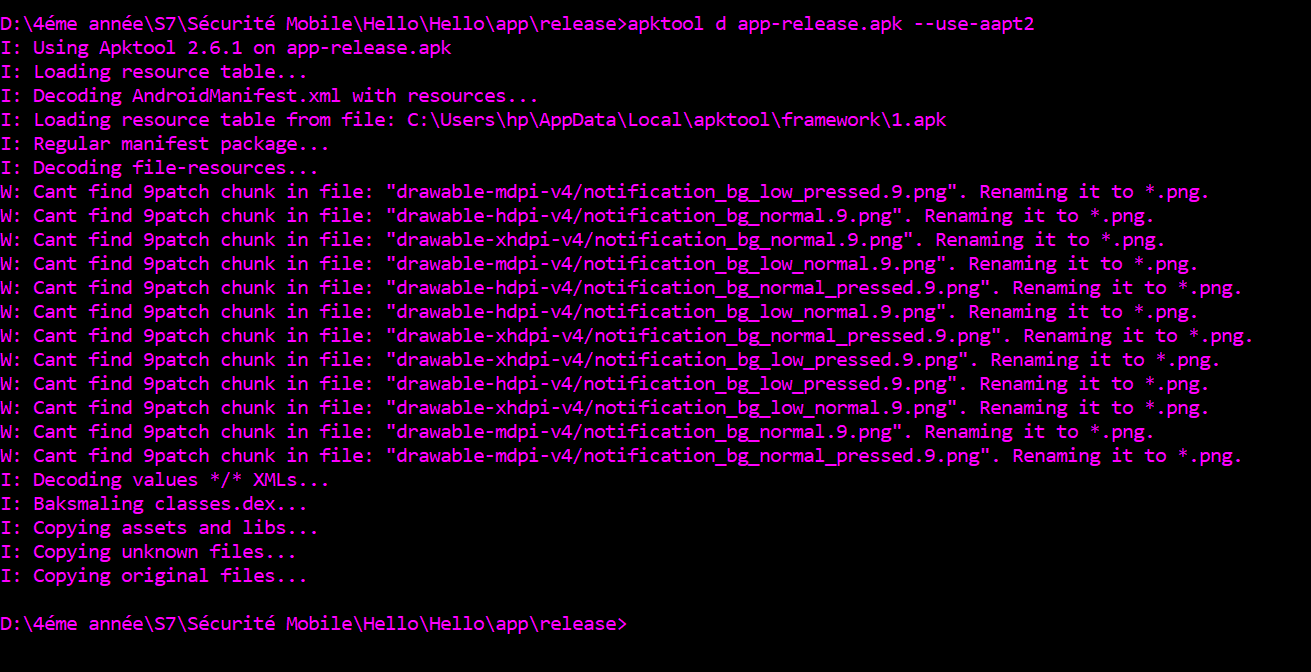
Displaying the logcat of my application after running it on the emulator:

As you can see here, inside our logcat, I can see the password.

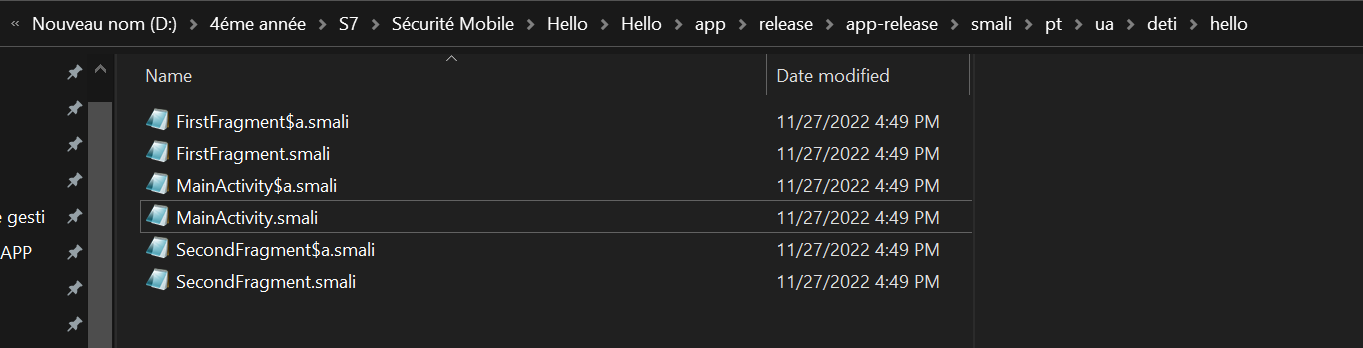


Therefore, I am going to navigate into my app release folder in order to get app-release.apk and apply some changes to it.

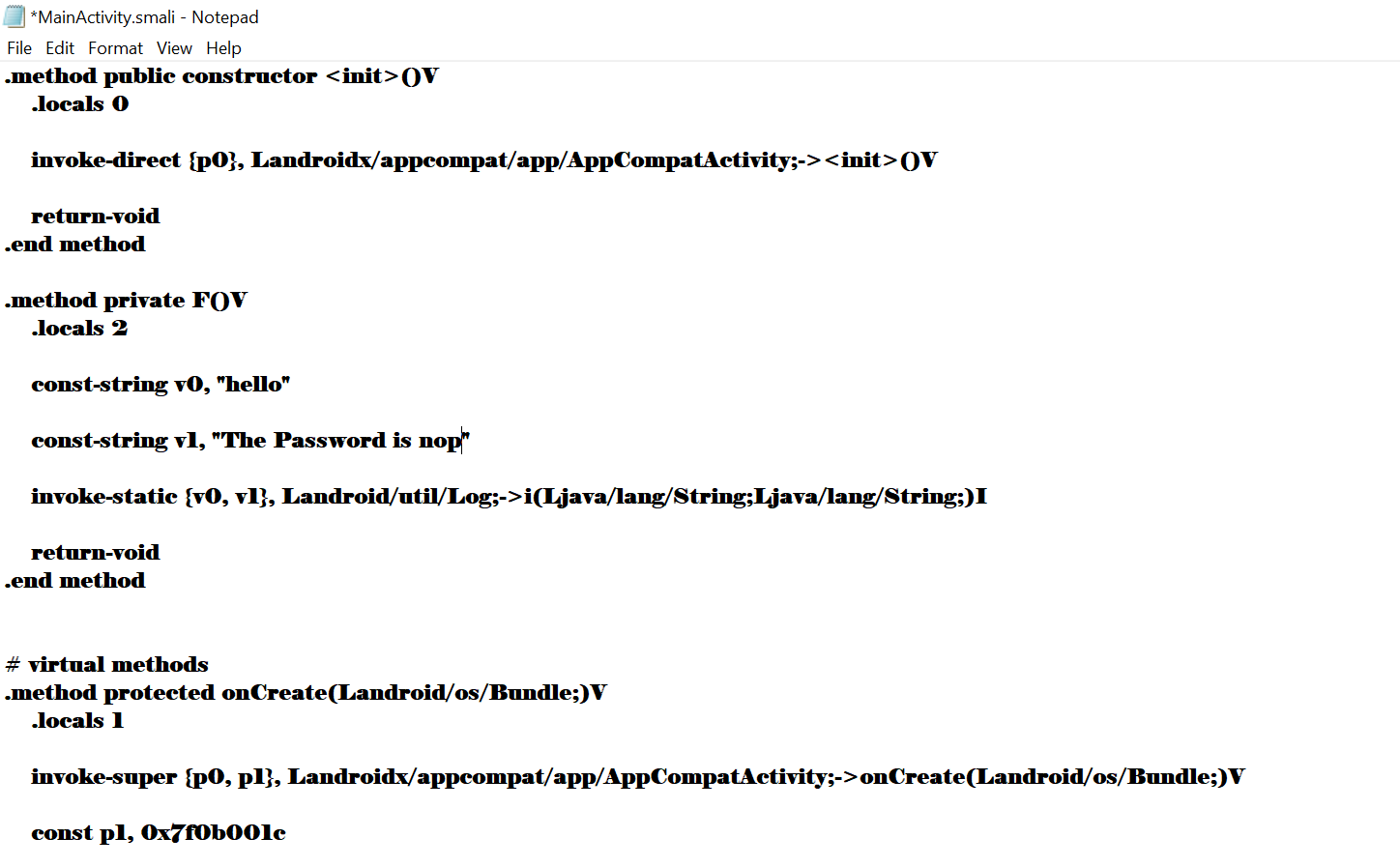
In the capture below, I’ve decompress the app-release.apk app:



And fond the activities inside this path in the decompressed folder:

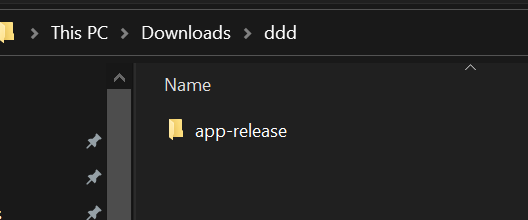


I opened « mainactivity » with notepad and applied some changes in order to hide the password

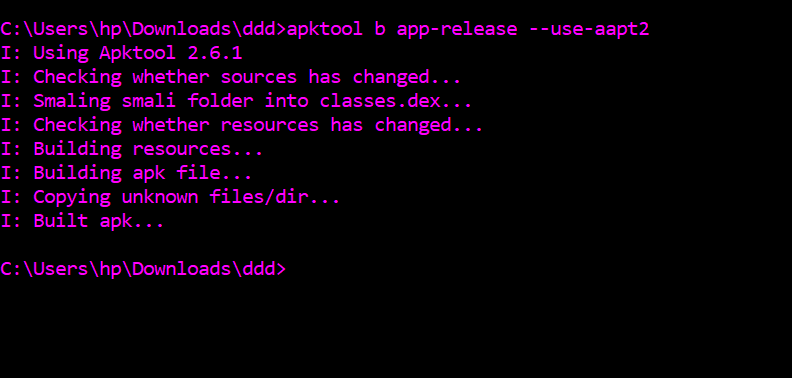


Replaced the password with <nop>

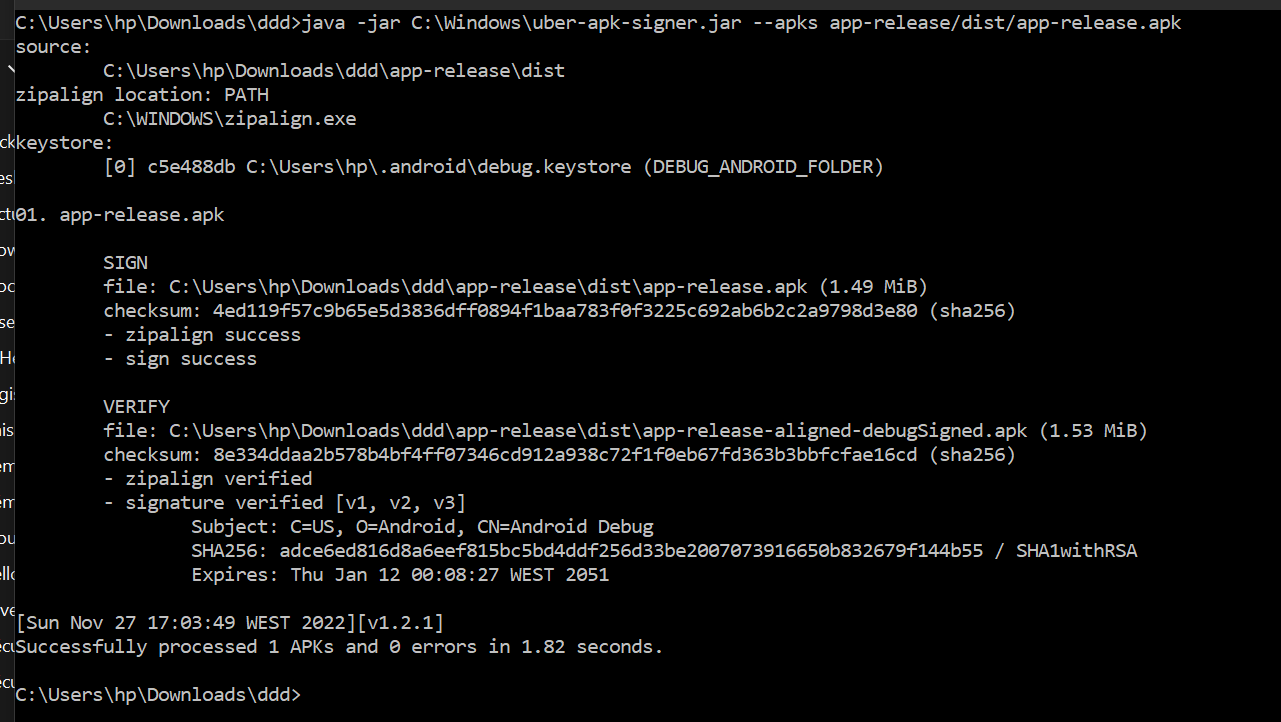
Then I’ve placed my app-release folder inside a new folder I created:



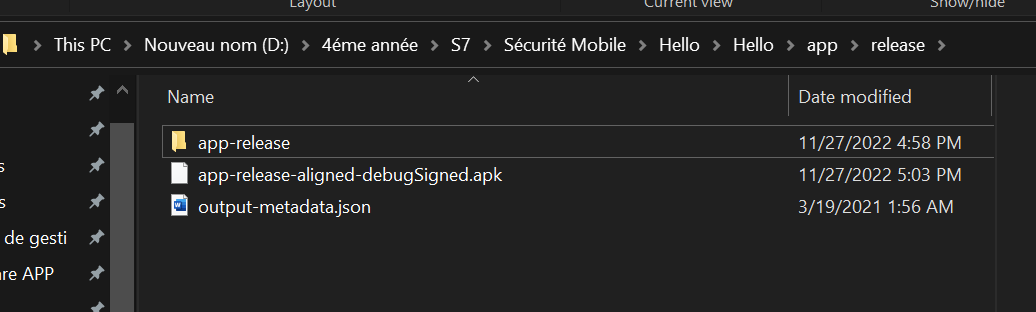
And used the apktool in order to regenerated a new Apk that has the changes:



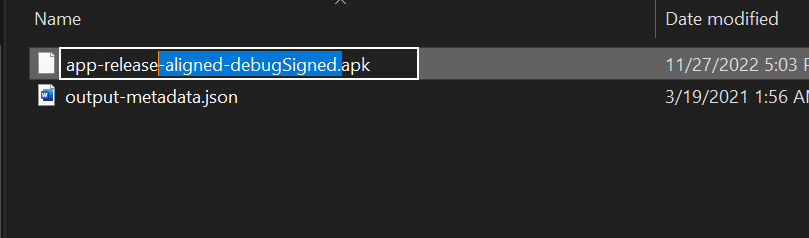
Then I’ve signed it using uber signer tool:



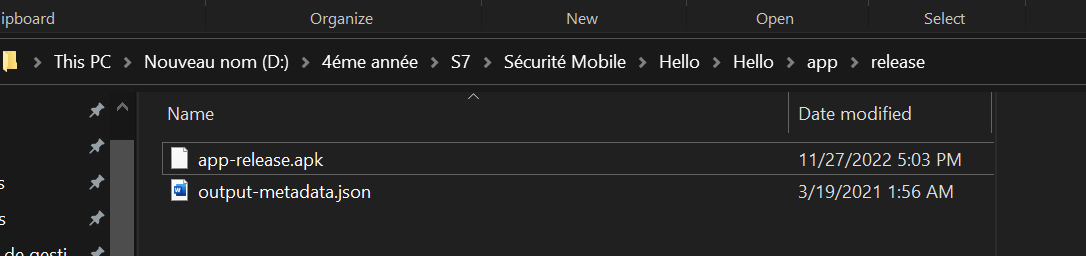
Then I’ve placed that signed Apk inside the release folder:



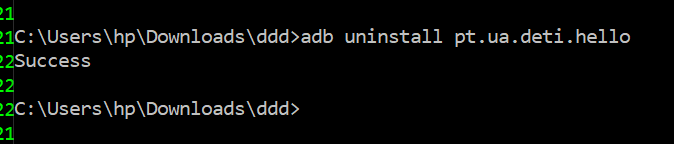
Decicded to rename it :



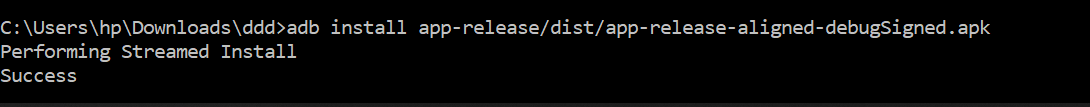
So that it takes place of the original Apk:



And I’ve uninstalled the previous package:



And downloaded the new modified Apk:



As you can see now, the password is no longer visible.

