1 TRUSTED CONTAINER FOR VIRTUALIZZATION

1.1 Composition

- Apk to analyze (GuestApp)
- Transformer: The tool is responsible for transforming (i.e., protecting) the input apk file.
  Modules:
  - Extractor => extract in a pseudo-random way some methods of the original apk, save their content in file.dex files and replace it with a Runtime-Excepton,
  - Protector => encrypt in a pseudo-random way some method and retrive the correct decryption key leveraging the comunication between container and plugin app.
  - Injector => add in a pseudo-random way some crontrols (e.g., AT, AD, etc..) inside the apk (NB: define which controls will be implementend and why).
- Container with VirtualApp framework that will start the guest app only after certain checks.

1.2 Transformer + Container VirtualApp

1.2.1 Transformer

1. Place the apk in any folder you like, preferably an empty one. For now the tested ones were puzzle_1.22.apk, thermometer.apk and telegram.apk
2. Open the Transformer file with Intellij and not Android Studio (e.g. the file does not have a Manifest.xml) and go to the **build.gradle** in which the arguments to be passed are these ones.

```
args "--android-jars", "path\_platforms\_SDK"
args "--output-path", "path\_output\_file"
args "--apk-path", "path\_apk"
args "--extractor-chance", "10"
args "--run-injector", "true"
agrs "--jarsigner-path", "path\_jarsigner\_path" (si trova nella
    dir bin del jdk jarsigner.exe)
args "--run-protector", "true"
```

   Please remember to comment args help which would not start the program

3. Always inside the file **build.gradle** in dependencies there is **implementation files**. Enter the correct pattern. As in my case just change the username to your own
4. Check **Settings->Build,Execution,Deployment–>Gradle** in **Gradle user home** there is the correct pattern that is dir **.gradle** corresponds to the project opened with Intellij.
5. BUILD: You are ready to build the project. However, it may not work and give this error:

```
Caused by:
    org.codehaus.groovy.control.$MultipleCompilationErrorsException:
    startup failed:"
```

To fix this you need to go to both **Project_Structure->Project** and **Settings->Build,Execution,Deployment–>Gradle** and change the version of the Gradle JVM and specific SDK module.The version of the Gradle JVM and specific SDK module I used are:

- In File -> Project Structure -> Project -> Android API 30 Platform version 11.0.10.
- In File -> Settings -> Build,Execution,Deployment -> Gradle -> Gradle JVM 1.8 java version "1.8.0_282"

It should give an exception if openjdk16 is used, so don't use it for now!!!

6. RUN: If the build was successful, then it's time to boot it up. To do this just go to the **Main.java** file and click on the green arrow next to the **public class Main** class to start it. Of course you'll see all the files in the destination folder where the apk resides.

7. Depending on the type of apk passed you can get different results. If you use an apk like thermometer.apk you may get an exception like this below:

```
Exception in thread "main" Exception in thread "Thread-17"
    java.lang.RuntimeException: Dex file overflow. Splitting not
    support for pre Lollipop Android (Api 22).
  at soot.toDex.MultiDexBuilder.hasOverflowed(MultiDexBuilder.java:96)
  at soot.toDex.MultiDexBuilder.internClass(MultiDexBuilder.java:58)
  at soot.toDex.DexPrinter.addAsClassDefItem(DexPrinter.java:670)
  at soot.toDex.DexPrinter.add(DexPrinter.java:1677)
  at soot.PackManager.writeClass(PackManager.java:1096)
  at soot.PackManager.lambdawriteOutput1(PackManager.java:699)
  at
      java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
  at
      java.util.concurrent.ThreadPoolExecutorWorker.run(ThreadPoolExecutor.java:624)
  at java.lang.Thread.run(Thread.java:748)
```

The reason is attributed to the fact that such apk has a minSdkVersion **less than 22**. So you need to set the apk with a minimum minSdkVersion to 22.

8. If the Apk, such as puzzle_1.22.apk and telegram.apk, has a version minSdkVersion of at least 22 then the previous hurdle is overcome, however we come to this exception:

```
Exception in thread "main" soot.CompilationDeathException: I/O
    exception while printing dex
  at soot.toDex.DexPrinter.print(DexPrinter.java:1706)
  at soot.PackManager.writeDexOutput(PackManager.java:585)
  at soot.PackManager.writeOutput(PackManager.java:567)
  at Main.main(Main.java:93)
Caused by: java.util.zip.ZipException: zip file is empty
  at java.util.zip.ZipFile.open(Native Method)
  at java.util.zip.ZipFile.<init>(ZipFile.java:225)
  at java.util.zip.ZipFile.<init>(ZipFile.java:155)
  at java.util.zip.ZipFile.<init>(ZipFile.java:169)
  at soot.toDex.DexPrinter.printZip(DexPrinter.java:294)
  at soot.toDex.DexPrinter.print(DexPrinter.java:1699)
  ... 3 more
Caused by: java.util.zip.ZipException: zip file is empty
```

The problem is in the options in the build.gradle in particular the option **args "–output-path", "path_output_file"**. For now to fix the problem just comment on it.

9. The output is in the sootOutput folder and will be in the same path of the apk.The execution time depends both on the HW and the size of the apk. The apk used by me to explain this report is **puzzle_1.22.apk**.

10. Another exception it might give is given by Jarsigner which is not found. Remember to enter the correct path in build.gradle. The Jarsigner is used to perform two fundamental tasks in Java archive security: signing a JAR file and verifying the signature and integrity of the file itself.

1.2.2 Container VirtualAPP

Once the Transformer process is complete, there are the following files in the sootOutput folder in the same dir as the apk:

- integrityCheck: with the integrity check information needed by the trusted container.
- signatures: with the mapping between the extracted method and the output folder
- Different dir: with strings formed by different letters and numbers. The bigger the apk the more such dirs there will be. Each dir corresponds to an extracted method. Inside of each dir there is classes.dex file.
- protectorDetails: with the information of the correct decryption key of the encrypted methods in the plug-in app. The trusted container is responsible for reading this file and restoring the correct decryption key.
- injectorDetails: with the information of the injected AT controls that will be hooked by the trusted container to complete successfully
- The protected apk is also present. You can see this because between the original apk and this one, the latter has a slightly larger size.

To securely launch the apk you must first transfer the files inside the asset folder of the TrustedContainerVirtualApp project, like this.

- Copying sootOutput/integrityCheck in asset/integrityCheck
- Copying sootOutput/protectorDetails in asset/protectorDetails
- Copying sootOutput/injectorDetails in asset/injectorDetails
- Copying the protected apk into asset
- Create a new folder named after the app's package name (for the puzzle_1.22.apk apk the package name is **br.com.cjdinfo.puzzle** and copy the signatures file and all folders with long strings into it.
- Open ContainerVirtualApp con Android Studio 4.1.
- It may give an exception during the build related to the NDK. To fix it **File-> Project Structure -> SDK Location -> Android NDK location** insert the path to the NDK. The one I used is

  ```
  C:\Users\claud\AppData\Local\Android\Sdk\android-ndk-r20b
  ```

- Go to **MainActivity.java**

  ```
   Helper.writeFileOnInternalStorage(this, "test.apk",
                  "pluginApps", Helper.readBytesFromAsset(this,
                      "puzzle_1.22.apk"));
  ```

  Instead of **puzzle_1.22.apk** insert the apk given as input to the Transformer

- Tried to start telegram.apk but the installation time was too long so it had to be stopped.
- Now you can run the project. I used,as emulator, Android 8 both x86 and x86_64 architecture.