IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 7, Issue 4, April 2018

Reverse Engineering APKS for Analysis

Sagar Lachure[1], Umesh Pagrut[2], Nilima Jichkar[3], Nilofer Khan[4], Jaykumar Lachure[5]

YCCE Nagpur[1,3,4]

SSCACS Akola[2]

GCOE Amravati[5]

Abstract: The Android operating system has a large number of user base worldwide which results in approximately 80 percent of the world population using android. The application it runs are called Android Package Kit (APK) which can directly be installed on android device for running the application. We present how these Application Kits can be reverse engineered for code inspection or for other general analysis purposes. We use old school method of decompressing the package as a file and then extract codes from it.This later can be passed on to third party tools for the analysis of the structure.

Keywords: APE, SDE (Software Developers Eit), XML (Extensible Markup Language), Obfuscation , Java, Android, App, API (Application Programming Interface).

INTRODUCTION

Mobile devices have become a crucial part and an essential tool for human life. The reason behind this can be the applications running on the device which makes every task easier. There are a lot of applications maybe in millions .Then such cases result in malicious practices done by the developers or attackers. As Android has the largest market share among mobile operating systems [1], most malware is developed for Android as well. The rate with which new malware appears in the wild increases by the year [2].

There are abundant tools available in the market for automatically testing the application or analyzing it. However the rates of it reporting false positives is way too high and everytime we cannot rely on such tools. Dynamic Analysis can be done as a part of testing it.At some points Automatic checking can be done when the data is in mass quantity.But for a thorough checking Analysing the code is the best thing to do.To get the working of the application,its structure,templates etc.,This method seems efficient.Some of the times the abrupt behaviour of the application gives the final resort to the analyst to only rely on manual labor i.e. Code Review and Inspection at such times techniques like these come in handy.Testing has always been a complex part when it comes to big applications.Some problems can be determined by Automatic Testing while some can only be checked by Manual Testing.Reverse Engineering apks provides a quick way of extracting native java code from the application so the analyst can read it and inspect it. Before we further dive into the actual process there are some basic components we need to know about the structures of the application which will make it easier for us to co-relate with the process of reversing it.

I. STRUCTURE OF AN APK

An .apk file is actually a zip compression package, which can be easily decompressed with decompression tools. The following is what we can see after the helloworld.apk file is decompressed using any Zip utility. We can see that its structure is somewhat similar to that of the new project.

```
|-- AndroidManifest.xml
|-- META-INF
| |-- CERT.RSA
| |-- CERT.SF
| `-- MANIFEST.MF
|-- classes.dex
|-- res
| |-- drawable
| | `-- icon.png
| `-- layout
|           `-- main.xml
`-- resources.arsc
```

Manifest File

AndroidManifest.xml is a required file for any application. It describes the name, version, access rights,

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 7, Issue 4, April 2018

referenced library files, and other information [,] of the application. If you intend to upload the .apk file to Google Market, you need to configure this .xml file. Here we skip it since there are too many references on this topic on the Internet.
The AndroidManifest.xml contained in the .apk file has been compressed. It can be decompressed using AXMLPrinter2 [,]. The command used is as follows:
java -jar AXMLPrinter2.jar AndroidManifest.xml[3]

META-INF Directory
META-INF Directory, where signature data is stored, is used to ensure the integrality of the .apk package and system security. When using eclipse to create an API package, a check calculation is performed against all files to be packed.[3]
The calculation results are stored in the META-INF directory. When installing an .apk package on OPhone, the application manager will implement the same procedure above. If the result is different from that under the META-INF directory, the system will not install the .apk file. This helps ensure that the files contained in the .apk package will not be replaced randomly. For example, it is basically impossible to replace any picture, code fragment, or copyright data in the .apk package by directly decompressing the file, replacing such content, and then repacking it. Therefore, this may protect the system from virus infection and malicious modification, increasing the system security.

Classes.dex File
Classes.dex is a java byte code file generated after the compilation using java source codes. The Dalvik virtual machine used by Android is not compatible with typical java virtual machine. Therefore, the structure and opcode of .dex files are different from .class files. All java decompilers available now can not process .dex files.
Android emulator provides a decompilation tool, dexdump, which can be used to decompile .dex files. First, start the Android emulator, and upload the .dex file into the emulator through Adb push. Find the .dex file after logging in through Adb shell. Implement the following command: dexdump xxx.dex.[3]

The results show that 6 class files (class0 to class5) are found, corresponding to the number of the .class under the directory /bin. We can tell from this that all .class files are included in the .dex file. However, it is very hard to find out the modification that has been made from the decompilation results of hello.java in Class #5. So, how to output "Hello, OPhone" becomes a problem. If the decompilation of branch jump table is incomplete, the dump is also incomplete. This problem also exists with fill-array data table. Besides, there are many other disadvantages. In a word, the decompilation results of dexdump are hard to read.
Dedexer is another tool available on the Internet at present. It can read .dex files with output in assembly-like language. The output is similar to jasmin[ ] output, but includes Dalvik byte codes. More details about Dedexer will be given in the next section.

Res Directory
Res directory is used to store resource files. For details about resource manager in .apk files, please refer to related articles on the OPhone SDN website.

resources.arsc
It is a binary resource file after compilation.[3]

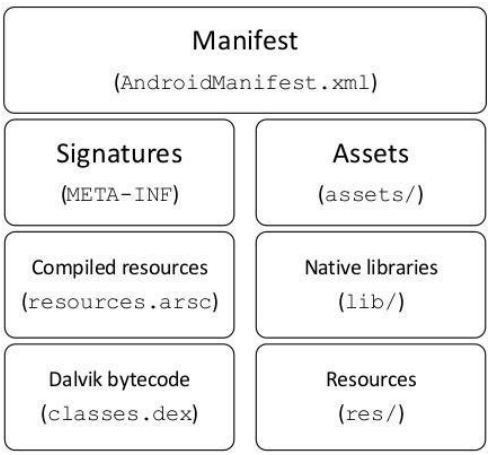


Fig.1 : Structure of an apk file[4].

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 7, Issue 4, April 2018

The classes.dex file from the structure is the most important file as it contains all the java code encapsulated and encoded in it.But dex files cant be read normally as they are encoded we require some special third party tools for it.

.

II. THE PROCESS

As we have seen above that the apk file is nothing but the compressed package of resources used by the android operating system to install an application.So the process starts with decompressing the apk.Which can be done in many ways. That is it can be done with a tool or without one.As the windows does allows changing file extensions therefore it can be done without requiring any tool

1. Go to the directory of .apk and Rename the apk to .zip and thus you are able to change the extension.
2. Now this compressed file can be extracted using stock tools or downloaded ones by right clicking on it and then clicking extract.

This will give you all the files shown in Fig.1 viz. Classes.dex META-INF etc. These files can be copied to a separate workplace of yours or some private directory. As we said earlier the most important file which is the classes.dex can be seen in the structure stores all the java code bundled together in it.For decoding it we require any third party dex to java converter which will help us extract java code from it. Dex2Jar[5] is a popular tool used by may people to do the same.

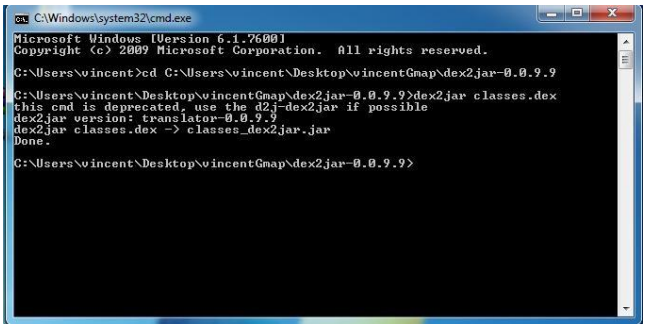


Fig.2 Working of Dex2Jar.

We have to just pass the .dex file which is extracted to the dex2jar.exe and then the tool decompiles it into a .jar file which a compress file of the java code and classes.
Usage: >dex2jar.exe classes.dex
The .jar however is not viewable yet.The class files in it are still encoded which can only be viewed by some other Java viewing third party tools.

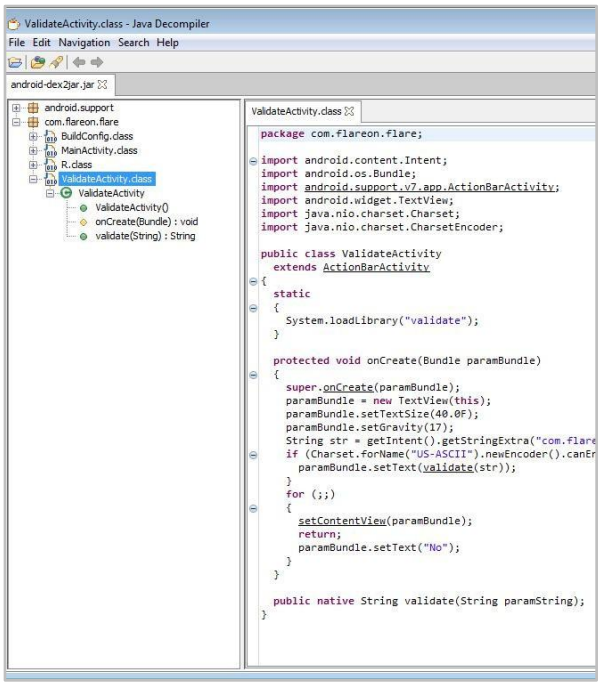


Fig.3 Opening class files in jd-gui

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 7, Issue 4, April 2018

Jd-gui is one of the tool which allows us to view the .jar files in a detailed and hierarchical manner.The jar file can be dragged and dropped for viewing otherwise for CLI users

Usage: >jd-gui.exe <yourfile.jar>

Up till now all the java files have been extracted and can be viewed using the jdgui utility as mentioned above.But Since Android APK or application not only consist of Java code butAlso XML data so for extracting information about layouts and interface we can use another tool provided by the Android SDK itself.The tool is "aapt" .It can be found in the build-tools directory of the android SDK.While building the projects XML files are not encoded enough so you can directly pass it the apk and it extracts the XML info from it.

Usage:>aapt yourapp.apk

This will generate a huge information in the console which can be copied to a separate file to view later .

III. PREVENTION TECHNIQUES

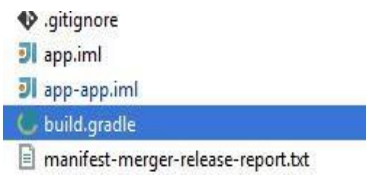
The decompilation of the apks can literally give you the Blueprints of any application.Viewing the Java code of a closed source project is pretty bad thing to think about.Thats why some developers prefer obfuscation of the code.

Code obfuscations transform a program so that it is more difficult to understand, yet is functionally identical to the original. The program must still produce the same results, although it may execute slower or have additional side effects, due to added code. There is a trade-off between the security provided by code obfuscation and the execution time-space penalty imposed on the transformed program.[6][7]

The process to do obfuscation can be done through Android IDE itself by setting up the property in the gradle configuration file.

Basic Obfuscation

To obfuscate code in Android studio just go to your build.gradle file in your Android Studio project:



IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 7, Issue 4, April 2018

Change the minifyEnabled property from false to true

```
buildTypes {
    release {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('proguard-android.tx
    }
}
}
```

```
buildTypes {
    release {
        minifyEnabled true
        proguardFiles getDefaultProguardFile('proguard-android.txt')
    }
}
```

This will take effect while the build of the application is done. Setting this to true might affect the build time therefore only official releases should go through this otherwise developers get a hard time debugging the application.[8]

Other than this there are open source tools available on the internet which help you obfuscate your code.

IV. CONCLUSION

Sometimes It is okay to follow the method of security through obscurity as shown here.Some threats are not discovered until they are known by the people other than developers.So Obfuscation might be a good standard to follow while developing projects but on the other hand.It can be a real nuisance while debugging the application.Also it does not totally guarantee the discreteness of the working of your projects. We all see the platforms getting flooded with new applications everyday but do we really know what is going on beside that chunk of code written.There are two sides to coin always.On the good side Miscreant works can be uncovered using techniques such as Reverse Engineering.But offensively thinking about it one can gain unauthorized access to APIS if able to understand the structure properly by doing the same.

REFERENCES

[1] Kantar. Android returns to growth in europe's big five markets. whitepaper, 2015.
[2] Pulse Secure. Mobile threat report 2015. whitepaper, 2015.
[3] "The Structure of Android Package (APK) Files". OPhone SDN. OPhone Software Developer Network. 17 November 2010. Archived from the original on 8 February 2011.
[4] https://www.researchgate.net/figure/Figure-APK-file- structure_fig2_316793316.
[5] https://github.com/pxb1988/dex2jar.
[6] Protecting Java Code Via Code Obfuscation, ACM Crossroads, Spring 1998 issue
[7] Mateas, Michael; Nick Montfort. "A Box, Darkly: Obfuscation, Weird Languages, and Code Aesthetics" (PDF). Proceedings of the 6th Digital Arts and Culture Conference, IT University of Copenhagen, 1–3 December 2005. pp. 144–153
[8] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan and K. Yang. "On the (Im)possibility of Obfuscating Programs". 21st Annual International Cryptology Conference, Santa Barbara, California, USA. Springer Verlag LNCS Volume 2139, 2001.