# A Study on Tools for Android Malware Analysis

Ali Almokhtar*, Dong-Hyun Kwon*, Yun-Heung Paek*
*Dept. of Electrical and Computer Science, Seoul National University
email : aalmokhtar@sor.snu.ac.kr

**Abstract**

Malware Analysis tools are being main topic research for many mobile security companies, in this survey, we are trying to go through the most popular tools used to find out the malicious codes and suspected android programs through reverse engineering process. There are so many malware tools have been made and implemented and some of them are efficient enough and others are quite slow and consuming high processing, however we are going to compare briefly some of them.

## 1. Introduction

Android is open source mobile operating system, it's based on Linux kernel and it is developed by Google. As the number of smart phones grows, Mobile malware is massively becoming threat and more serious, and because the huge reason the number of malwares is rapidly increasing due to the open source programs and open market for the applications plus most of the phone companies use Android as their operating system, there should be some way to defend and mitigate the malwares. In Figure 1, Kaspersky security company shows that android systems have the most number of mobile attacks compares to J2ME, Symbian, Windows and others. There are many Android malware analysis tools. These tools are used in various Android malware detection papers[11][12][17][18][19]. But each of them have very different functionalities. For example, they can provide dynamic analysis, static analysis, visualization of control flow and etc. So in this paper, we summarize each Android malware analysis tools and compare them**.**

In chapter 2 we present about the background and the terms are being used in the paper, in chapter 3 we present comparison between malware tools and how does each on work, and in chapter 4, we present the conclusion of our work.

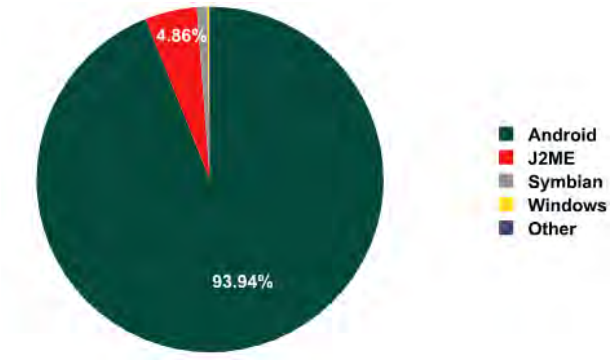



**Figure 1 Mobile malware detected 2004-2012 by platform[6]**

## 2. Background

Android application is distributed as APK(Android application package) file format. As shown in Figure 2, APK file is actually archive file. It consist of several files and directory, such as *AndroidManifest.xml*, *classes.dex* and *lib* directory[20]. *AndroidManifest.xml* file describe the name, version number, permission information and etc. *classes.dex* file include class files which compiled from Java source code. The *lib* directory contains native binaries. So APK file contains a lot of information about the application. Using this feature, android malware analysis tools usually extract information from APK file to detect malware.

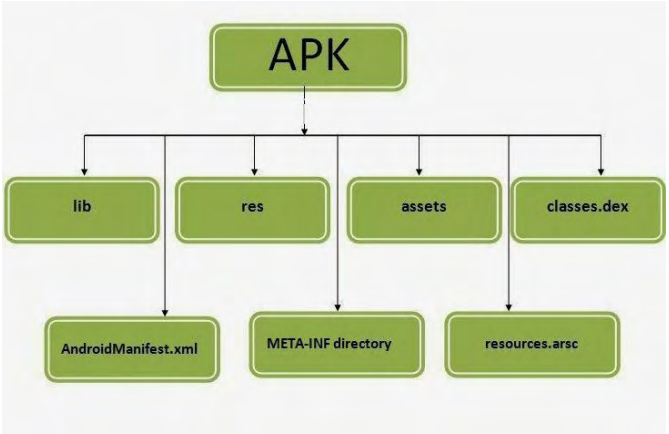



**Figure 2 The structure of APK file[21]**

## 3. Tools for Android Malware Analysis

### 3.1 Androgaurd

Androgaurd is Android analysis tool written by Python to disassemble and decompile APK file. Unlike with other tools, Androgaurd has the ability to check if the application is listed in the open source database of malwares so it will notify whether the application has been listed the black listed database. It can also measure the efficiency of the obfuscators and provide visualization functionality of output.

### 3.2 Andriod-apktool

Andriod-apktool is a tool for reverse engineering, it's also capable to decode resources to original form and repacking them after having some modifications, there is also possibility to analysis and debug smali code line by line. Smali code is assembly language based on Jasmin syntax which has full functionality dex format[8].

### 3.3 Dex2Jar

Dex2jar is designed to read dex(Dalvik Excutable) file format of Android application into jar format which is understandable and readable by other java reversing tools,

more ever the dex2Jar is used by APKinspector to transform the jar file into understandable format

Dex2jar contains of 6 components:[2]

1- *dex-reades* : is designeed to read the Dalvik Excutable format.
2- *dex-translator* : is designed to do the convert job and it reads the dex instruction to dex-ir format after some optimize after that convert to ASM format,
3- *dex-ir* : used ti dex-translator to represent the dex instruction,
4- *dex-tools* : tools to work with ,class files like modify a apk and DeObudscate a jar
5- *d2j-smali* : is to disassemble dex tosmali files and assemble dex form smali files
6- *dex-writer* : is to write dex same way as dex-reader

### 3.4 Dexter

Dexter is a static web based malware analysis that allows uploading android applications which needs to be analyzed. The tool extracts as much information as possible from either legitimate or malicious applications (APKs) and displays them in various different views[3]. It shows a quick overview of all metadata and included packages of the application, more ever the dependency graph shows all included packages and its interconnections with ability to go through all the method list of each method and each method will show list of classes and functions.

### 3.5 APKInspector

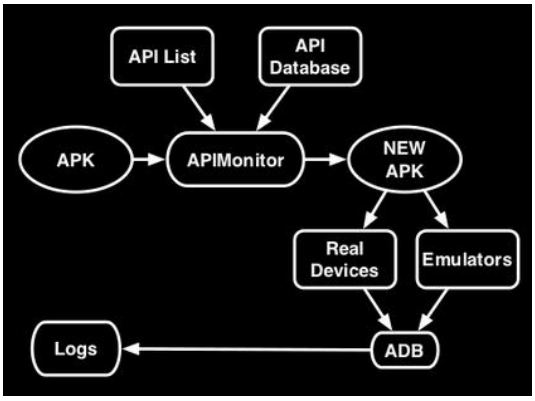
Apkinspector is a group of tools in one user interface. After the .apk has been loaded you can load the Smali representation of functions by selecting the function in the Methods tab in the side view. APKInspector comes with Jad[14], a Java decompiler. It should be able to decompile most classes, but regularly creates mistakes that either prevent a recompilation or sometimes make the class very hard to understand. Also it might fail completely in some cases, then the smali representation must be used[4].

There are new analysis features have been developed recently[10]:

- Reverse the Code with Ded[13] for Java Analysis
- Static Instrumentation
- Combine Permission Analysis

### 3.6 API monitor:

API Monitor is an open source software that allows you to monitor and control API calls which are revoked by the applications and services, it's one of the most powerful applications to track down and analysis the problems occurred during running in android application, and it supports 32 bit and 64 bit applications, there are more than 13000 API's definitions from 200 DLL's plus over 17000 methods. It also decodes and displays 100 different types of unions and structures. It has the ability to display tree which explains the hierarchy of API calls and the duration, call stack for each call.




**(Figure 2)** APIMonitor Architecture[7]

## 4. Conclusion

There are so many malware android tools have been implementing in many researches, and each one of these has its own specifications and drawbacks. Also each one varies on the time consumed to transform the Dalvik executable files into jar files, for example Androgaurd has many other tool compare to other malware analysis tools and it has the ability to visualize the application with gephi or cytospace or PNG/DOT format. More ever it can determine if the application has been pirated or altered, and also it can show indicators if the application might have malicious code or even suspected. Most of tools transform the APK to Java. They, however, do not provide reverse engineering tools for compiled part of APK file. So, in the future work, Android malware tools to implement full reverse engineering of Android application should be able to hack native binary of APK.

**Acknowledgement**

This work was supported by the Engineering Research Center of Excellence Program of Korea Ministry of Science, ICT & Future Planning (MSIP) / National Research Foundation of Korea (NRF) (Grant NRF-2008-0062609). This work was partly supported by the IT R\&D program of MSIP/KEIT [K10047212, Development of homomorphic encryption supporting arithmetics on ciphertexts of size less than 1kB and its applications], This work was supported by the Brain Korea 21 Plus Project in 2014 and IDEC, This work (Grants No. C0218072) was supported by Business for Cooperative R&D between Industry, Academy, and Research Institute funded Korea Small and Medium Business Administration in 2014

**References**

[1]http://resources.infosecinstitute.com/android-malware-analysis/
[2] https://code.google.com/p/dex2jar/
[3] http://dexter.dexlabs.org/static/docs/
[4] www.exploit-db.com/download_pdf/33093
[6] http://securelist.com/analysis/publications/36996/mobile-malware-evolution-part-6/
[7]http://www.slideshare.net/KelwinYang/improving-droidbox
[8] https://code.google.com/p/smali/
[9] http://www.xda-developers.com/tag/apktool/
[10] https://code.google.com/p/apkinspector/

[11] Eder, Thomas, et al. "ANANAS-A Framework For Analyzing Android Applications." *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on. IEEE*, 2013.
[12] Xu, Junfeng, Shoupeng Li, and Tao Zhang. "Security Analysis and Protection Based on Smali Injection for Android Applications." *Algorithms and Architectures for Parallel Processing. Springer International Publishing*, 2014. 577-586.
[13] http://siis.cse.psu.edu/ded/
[14] http://varaneckas.com/jad/
[15] http://gephi.github.io/
[16] http://www.cytoscape.org/
[17] Wu, Dong-Jie, et al. "Droidmat: Android malware detection through manifest and API calls tracing." *Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference on. IEEE*, 2012.
[18] Xu, Rubin, Hassen Saïdi, and Ross Anderson. "Aurasium: Practical Policy Enforcement for Android Applications." *USENIX Security Symposium*. 2012.
[19] Tchakounté, F., and P. Dayang. "Qualitative Evaluation of Security Tools for Android." *International Journal of Science and Technology* 2.11 (2013).
[20] http://en.wikipedia.org/wiki/Android_application_package
[21] http://pavan2pyati.blogspot.kr/2013/12/android-apk-file-format.html