A Survey on Detection Techniques of Android Malware

Nikita Rai* and Dr Tripti Arjariya#
nikita.rai18@gmail.com * tripti.beri@gmail.com#

***Abstract*—Android market is growing at rapid speed to become the Information base for diverge fields. Unfortunately the crimes over the Smartphone are also increasing at much faster rate with high complexity for illegitimate benefits. Presence of vulnerability and large attack vector in the android operating system change the face of clean application to malicious one. Malicious applications are collection of Trojan, spam, viruses and different type of spyware. These types of applications are threat to the basic security and are responsible for the vulnerabilities to mobile devices.**

Detection of malicious applications is one of the searing research topics in the field of security. It is also important because it prevents the novice end user from being compromised from attacks on Smartphone. This paper takes a closer look at techniques to detect the mobile malwares.

Keywords— Android, Android Malwares, Smartphone, Trojan, Spyware, Manifest File, APK File

I. INTRODUCTION

Smartphone are the widely used mobile devices in which novice end user saved sensitive information, financial information and personal content like photos, videos, bank details etc. On the flip side a huge number of malwares are being exploited to steal the private information. Researchers are working on the detection of malwares in Smartphone to prevent the novice end users.

There are lots of smart phones available in market using different types of operating system. Android is one of the prevalent operating system for mobile devices in market. It is an open source operating system based on Linux kernel. Since 2008 Android is the most popular operating system for smart phone.

With the increase in market of the android, attackers stated targeting android devices such as Smartphone. Hence, exploitation of the Smartphone is the greatest target of the attackers. Attacker attacks on their devices to steel their information. Due to openness of the android system it becomes very easy for attacker to develop malware which can harm any android device.

This paper discussed the different methodologies used to detect and analyse the malwares in mobile applications. It mainly focused on three major techniques of malware analysis i.e. static analysis, dynamic analysis and hybrid analysis.

II. ANDROID MALWARE

Android malware is define as an android application which are developed with the malicious intension to steal personal information, financial frauds, affect the normal working of system and infrastructure, etc.

Usually the source of the android malware comes mainly from third party markets. Recent reports focusing on mobile malware trends [1] estimate that the number of malicious Android apps is now in the range of 120,000 to 718,000.

Mobile malware can be classified according to their functionality like: privacy escalation, remote control, financial change and private information stealing. Here, the android malwares are classified into two categories as Trojans and spyware; with respect to use of social engineering to trick end user into installing the malicious software.

A. SMS Trojans

The majority of android malwares are classified as SMS Trojans or Fake Installers [2]. These apps pretend to be an installer for legitimate software and trick end users into installing them on their devices. When executed, the app may display a service agreement. Once the novice end user agreed the service agreement, it sends premium rated text messages to attacker.

Variants include repackaged applications that provide the same functionality as the original—often paid—application, but have additional code to secretly send SMS messages in the background. SMS Trojans are relatively easy to implement: only a single main activity with a button that initiates the sending of an SMS message when clicked is required.

B. Spyware

Another observed type of Android malware is classified as spyware and has capabilities to forward private data to a remote server. In a more complex form, the malware could also receive commands from the server to start specific activities in which case it is part of a botnet.

Broadcast receivers are of particular interest as they can be used to secretly intercept and forward incoming SMS messages to a remote server or to wait for BOOT COMPLETED to start a background service as soon as the device is started. There are also lots of Spyware Applications which may steals the personal information and send it to remote server.

III. ANALYSIS TECHNIQUES FOR ANDROID MALWARES

Techniques for analyzing android malware may be classified into three types such as static analysis techniques, dynamic analysis techniques and hybrid analysis techniques.

A. Static Analysis Techniques

The main focus of static analysis of mobile malwares is to check the source code of the application

and manifest file of the application. Static analysis extract code and xml file from the APK file. In static analysis, contribution of the APK file is important because it contains the information related to the application. Code is analyzed to find out the obfuscation and suspicious code. Manifest file is analyzed to check the permission as well as resources and services of the application. Common static analysis techniques for malware analysis are as follows.

1) Signature based Static Analysis Technique

The signature contains the message digest of the APK file [3]. Since any modification to the APK file will change them message digest of the signature, one could quickly identify if an application is corrupted by checking the signature. Analyst could also collect signatures of malwares to find out malwares quickly.

2) Byte Code based Static Analysis Technique

The executable part of the application, the '*class's.dex file'* in the archive, contains all compiled classes of the program in the form of byte codes. For Android programs, the original JAVA byte codes are converted to the instruction set used by the Dalvik virtual machine (DVM), which is a register-based virtual machine.

3) Resources based Static Analysis Technique

Resources is the non-executable part of the application, it contains all additional data required by the application. Most resources in an application are user interface components, such as bitmaps, menus, layouts, widgets. In most cases, the malicious part of the malware runs in background and does not have any user interfaces. So these UI resources are seldom concerned.

However, the resource file such as *'AndroidManifest.xml*' is important because it indicates crucial forensic information of an application. The AndroidManifest.xml file is encoded into binary format in the APK file. It contains the permission request of an application. The most important forensic information is components and permissions.

4) Components based Static Analysis Technique

Android applications are formed by components. The components of the application are divided into four kinds – activities, services, broadcast receivers and content providers [2].

A malware that executes in background often has a service component and a receiver component in order to receive the boot Intent on system booting. Through checking components and their received intents, analyst may have a brief view of the potential behaviour of an application.

5) Permissions based Static Analysis Technique

In order to access some protected APIs of Android, the application will declare the permission request in AndroidManifest.xml, such as the permissions to read message, contacts, etc.

Permission request is a very important clue to reveal malicious functions. For instance, a normal application, such as calculator, declares a 'READ CONTACTS' permission, it can be very suspicious because a calculator should never need information about contacts. This character is unique for Android applications and is useful for analysis.

B. Dynamic Analysis

Commonly dynamic analysis techniques analysed the behaviour of the application. It monitors the file system, network flow, outgoing SMS, phone calls as well as native code during runtime [4]. Additional analysis like stimulation, taint tracking, method tracing and system level analysis.

There are lots of frameworks are available to analysis the mobile malwares on the basis of the behaviour of application. Some of the frameworks with their working processes are as follows.

1) Ananas Framework

It allows the implementation and integration of several different analysis methods into one powerful platform [5]. Ananas framework provides common services that are needed for the dynamic analysis. One service requirement of dynamic malware analysis is a clean and emulated environment. Architecture of Ananas is shown in Figure 1.

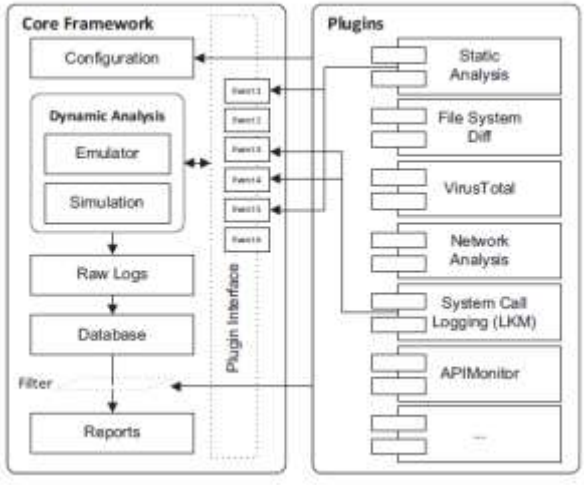

Figure 1: Ananas Framework

2) Taintdroid Framework

It is a multiple granularity taint tracking approach on android system [2]. Firstly, Taint Droid automatically taints all data from sensitive source and put labels as sensitive data. When these labelled data transmitted through the network or leave the system than Taint droid logs the data's labels and identify transmitting the data.

Taint droid only tracks data flow not control flow that means only explicit data can be trace not implicit data. It is very effective for tacking sensitive information but it causer signification false positive when the tracked information contains configuration

identifiers. Framework of Taint droid is shown in Figure 2.

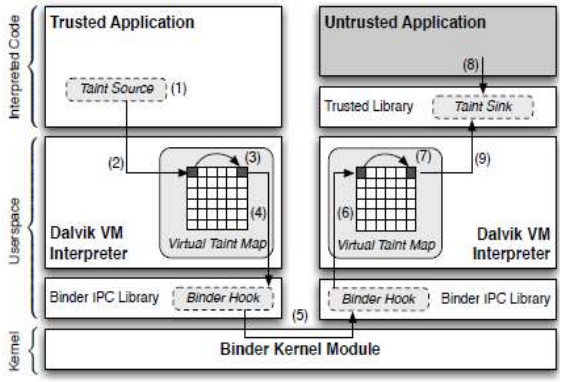

Figure 2: Taintdroid Framework

3) Droid Scope Framework

It performs dynamic taint analysis at the machine code level. With semantic knowledge at both OS and Java levels, Droid- Scope is able to detect information leakage in Java components, native components, or even collusive Java and native components.

To facilitate custom analysis, Droid Scope [6] exports three tiered APIs that mirror the three levels of an Android device: hardware, OS and DVM. It logs all the classes, fields and all available symbol information. Droid Scope's architecture is depicted in Figure 3.

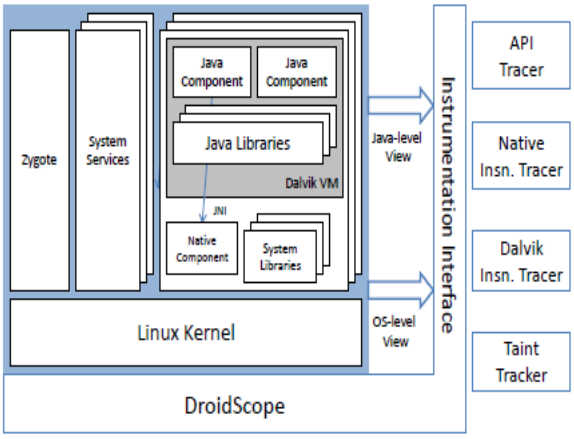

Figure 3: Droid Scope Architecture

Furthermore, some similar changes has been made to a different version of QEMU 3 to enable x86 support to demonstrate the capabilities of Droid Scope [7]. It monitors communication of malware's Java components with the Android Java framework, interaction of native components with the Linux system, and communication of Java components and native components communicate through the JNI interface.

4) Profiledroid Framework

Analysis of Profiledroid is work in 4 layers such as Static specification, user interaction, operating system and network [8]. All 4 layers consists two part first one is monitoring and second one is profiling. Monitoring system logs all the activity of the application and sends it to a specific computer for profiling.

The static layer analyse the APK file. It mainly focuses on manifest.xml files and the byte code files.

The user layer, focuses on user generated event, i.e., the event occur when a user interact with the android device at the run time. The OS layer analyse the operating system activity through monitoring the system calls which are invoked by the application. The network layer analyse network traffic through logging the data packets. Profiledroid Framework is shown in Figure 4.

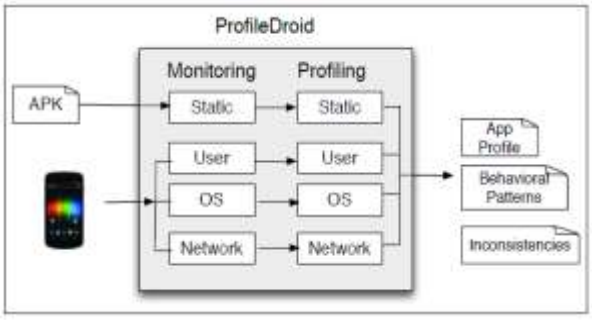

Figure 4: Profiledroid Framework

C. Hybrid analysis

In Hybrid analysis results of static analysis are used to guide dynamic analysis extend coverage of executed code. It is required when analyst is not able to find out the overall behaviour of the application through static and dynamic analysis separately.

The process of hybrid analysis is starts from the static analysis. Static analysis firstly checks the code obfuscation and then checks the permissions pattern. After that dynamic analysis checks the behaviour of application as well as IP call of the application.

There are only few some frameworks which work on both static as well as dynamic analysis, which are called as hybrid analysis framework. Moreover, there is no hybrid framework that dynamically monitors both actions within the DVM and outside it in native libraries.

1) Mobile Sandbox Framework

It is a novel hybrid system that may track native API calls and easily accessible through a web interface [9].

Within the static analysis part it analyse the application to get an overview of the application. First, need to perform several anti-virus scans, secondly, parse the manifest file, and finally decompile the application to better identify suspicious code.

Within the dynamic analysis, the application execute in an emulator and log every operation of the application, i.e., log both the actions executed in the Java Virtual Machine Dalvik and actions executed in native libraries which may be bundled with the application.

It also checks the native call as well as network

traffic for better understanding the behaviour of the application. Framework of Mobile Sandbox is shown in Figure 5.

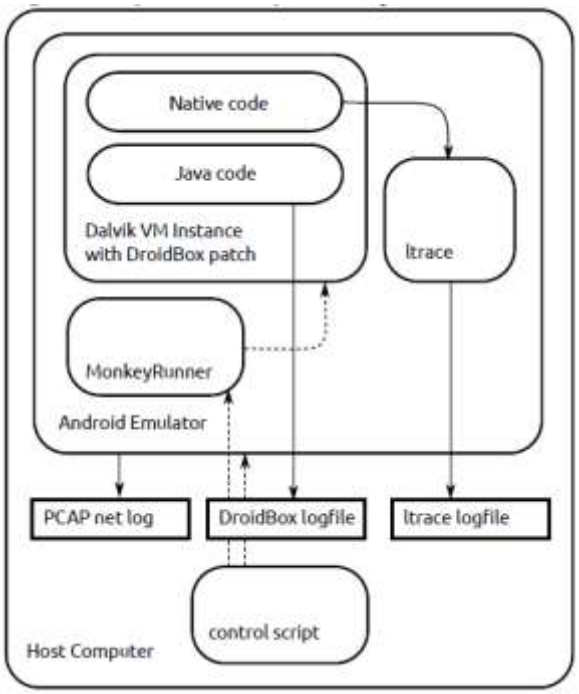

Figure 5: Mobile Sandbox

2) Andrubis Framework

It follows the hybrid analysis approach and results of the static analysis are used to perform more efficient dynamic analysis [10]. Components of Andrubis and their relation with each other are shown in Figure 6.

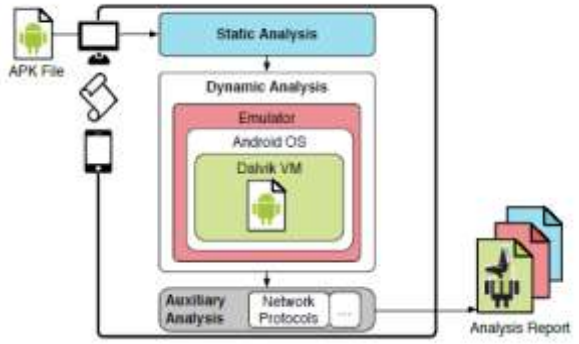

Figure 6: Andubis Framework

There are following analysis stages of Andubis Framework:

Static Analysis: During this stage extract information from an app's manifest and its byte code.

Dynamic Analysis: This method executes the app in a complete Android environment, and its actions are monitored at both the Dalvik and the system level [10].

Auxiliary Analysis: Capture the network traffic from outside the Android OS and perform a detailed network protocol analysis during post-processing.

IV. Related Work

To improve this situation, many researchers have tried to interpret Android permissions and their combinations.

M. Spreitzenbarth et Al. [11] developed a system to automatically analyse the android applications in two novel ways. The system is based on static and dynamic analysis techniques. Author uses the specie technique to log calls to native (i.e., \non-Java") APIs.

Juanru Li et Al. [12] proposed a systematic procedure for Android malware forensic analysis and malicious events reconstruction. Author in this paper discussed about to defeat anti-forensics codes and combine the existing techniques to help analysis.

Lei Cen et Al. [13] proposed a probability model to identify the common required permission patterns for all Android applications. In order to detect the malwares author extract the features from the Android .apk files.

Zhang et Al. [14] listed the top required permissions for both clean and malicious applications, but only individual permissions were considered by frequency counting.

Zarni Aung and Win Zaw [15] classify the android application as benign and malicious thorough applying the machine learning techniques on permission features.

Dai-Fei Guo et Al. [16] proposed a behavioural based approach to detect the metamorphic malwares on the basis of classification through self learning data mining. Author analyse the behaviour of metamorphic mobile malwares.

V. Conclusion and Future Work

Mostly novice end users are careless about to check the permission used by the particular application during the installation of application on mobile device. It became easy for any attacker to change the permission of the application to suspicious one.

Common frameworks check the code, behaviour and permission of the android application to analyse the nature of the application. Permissions are one of the access points of the android to access any desire component of the application. However, permission may play important role to detect the android malwares through identifying the permission which work as a malware.

In this paper it has been found that there are methodologies to identifying the permission pattern of the APK file. But due to rapid growth of the android application new permission are also generating which can be behave as a threat for any Smartphone. Finally the accuracy of the framework based on permission pattern should be quite effective to identify the android malware.

Attackers launched new different types of malware, for analysing these new malwares another approach is required which may easily identify it. Future work also

includes evaluating the more static analysis methods to minimize the false positive rate.

REFERENCES

[1]. Sonal Nerurkar, "Teens drive Indian smartphone sales, study finds" online[Available], http://timesofindia.indiatimes.com/ business/indiabusiness/Teens-drive-Indian-smartphone-sales-studyfinds/articleshow/22406572.cms [Accessed : 23 Nov 2013]

[2]. Cesare, x. yang and silvio, "Classification of malware using structured control flow," Australian Computer Society, vol. 107, pp. 61-70, 2010.

[3]. Spreitzenbarth, Michael and F. Felix, "Mobile-sandbox: having a deeper look into android applications," Annual ACM Symposium on Applied Computing, pp. 1808-1815, 2013.

[4]. "Apktool download," [Online]. Available: code.google.com/p/android-apktool/. [Accessed 11 03 2015].

[5]. Allix, kevin and q. jerome, "A Forensic Analysis of Android Malware--How is Malware Written and," compsac, pp. 384-393, 2014.

[6]. "Android malware hijacks power button," [Online]. Available: http://www.theregister.co.uk/. [Accessed 12 5 2015].

[7]. Machiry, Aravind and T. Rohan, "Dynodroid: An input generation system for android apps," in Foundations of Software Engineering, 2013.

[8]. Aung, Zarni and Z. Win, "Permission-based Android malware detection," International Journal of Scientific and Technology Research, pp. 228-234, 2013.

[9]. Barrera, David and G. H., "A methodology for empirical analysis of permission-based security models and its," ACM conference on Computer and communications security, pp. 73-84, 2010.

[10]. Min, X. Luo and H. C. Qing, "Runtime-based behavior dynamic analysis system for android malware detection," In Advanced Materials Research, vol. 756, pp. 2220-2225, 2013.

[11]. M. Spreitzenbarth et Al., "Mobile-Sandbox: Having Deeper Look into Android App.", SAC'13 March-2013, Coimbra, Portugal, pp. 18-22.

[12]. Juanru Li, DawuGu and YuhaoLuo, "Android Malware Forensics Reconstruction of Malicious Events", In 32th Int. Conf. on Distributed Computing Systems Workshops 2012.

[13]. Lei Cen et Al., "A Probabilistic Discriminative Model for Android Malware Detection with Decompiled Source Code", Pub. In IEEE Trans. on dependable and secure computing, Vol.12, Issue. 04, 2015, pp. 400-412.

[14]. Zhang, yuan and y. min, "Permission Use Analysis for Vetting Undesirable Behaviors in Android Apps," Information , pp. 1828-1842, 2014.

[15]. Zarni Aung and Win Zaw, "A Probabilistic Discriminative Model for Android Malware Detection with Decompiled Source Code", Int. J. of Scientific & Technology Research, Vol. 2, Issue 3, March 2013, pp. 228-234.

[16]. Dai-Fei Guo et Al., "Behavior classification based self-learning mobile malware detection", Pub in J. of Computers, Vol. 9, Issue 4, 2014, pp. 851-858.

