Hybrid Model for Detecting Android Malware

Sonia Sara James[1] Neenu Paul[2] Amala Baby[3]
[1, 2, 3]Amal Jyothi College of Engineering Mahatma Gandhi University Kerala, India

Abstract--- Android applications are commonly used in smart phones. Nowadays android phone automatically downloads applications without checking that it is a suspicious application or not. Applications downloaded from the internet may be malicious. Malwares can interrupt system operations and can affect the memory usage. The malwares generally appears in the form of signatures, codes, scripts etc. Signatures cannot be found on application before installation process. They are only found during execution. Through this paper we propose hybrid model for the detecting android malwares. Our application automatically detects the malware affected by it by performing analysis .Our application consist of a sandbox which provides an isolated environment. Hybrid model performs two types of analysis, the static analysis and dynamic analysis. Static analysis scans the malware before installing it. It reports the detected malware with the region affected by it. The dynamic analysis is a run time process. Dynamic analysis detects the malware affected by the installed APK (Android Application Package) file. Dynamic analysis is performed on android emulator. It generate corresponding log files, memory status and system performance level.

Keywords: Malicious software, Static analysis, Dynamic analysis

I. INTRODUCTION

The Android platform is being the fastest growing market today, facing risks in a large scale. Malware takes advantage of the android platform that its being an open source, complete and free for development etc. Allowing anyone to develop and to publish applications into the Android Market provides an opportunity for attackers to easily deliver malicious applications into unsuspecting users. The presence of malicious applications in the Android Market exhibits user's security from malware attacks. The most common Android suspicious application contains spyware and Trojans that performs malicious functions such as:
It collects and sends GPS coordinates, contact lists, e-mail addresses etc to third parties. They records phone conversations and send them to attackers. They took control over the infected phone. It downloads other malware into affected phones. Examples of some malware carrying apps are DroidDream, DroidKungFu, Geinimi, ANDROIDOS_NICKISPY.C etc

The Android platform uses permissions-based security models to have access to different functionalities of devices. This model provides information about the access and privilege capacity of an application which to more technical users, may be used as an indication for malicious intent but to normal users, this information is often neglected thus making this model unreliable on its own. Static analysis is a method used to detect malware but this method proves to be inadequate as malware can be undetected by obfuscation, as an example. The time it takes to manually check the code provides an opportunity for malware to infect devices before they are detected. The number of users is increasing and so is the number of devices on the market. This attracts malware authors to target Android devices with the intentions of economical profit, stealing private data and infecting android devices.

Figure. 1 shows the malware targeting android platform based on statistics in the year of 2010. The most commonly affected malware affected is android platform [7].

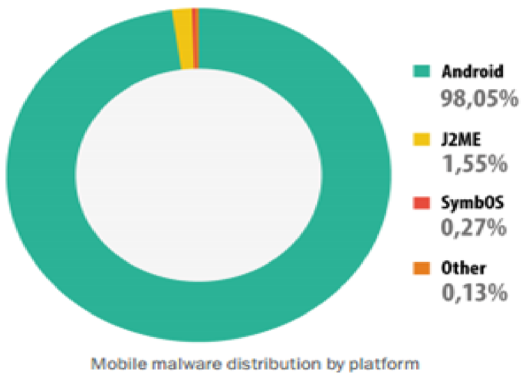


Fig. 1: Malware distribution by Android platform as on 2010.

The mobile malware found in the following case study [7] is explained in the figure2. The total number of mobile malware found is 148,778 of them were found in 2013.Still, the trend is highly visible and continuing.

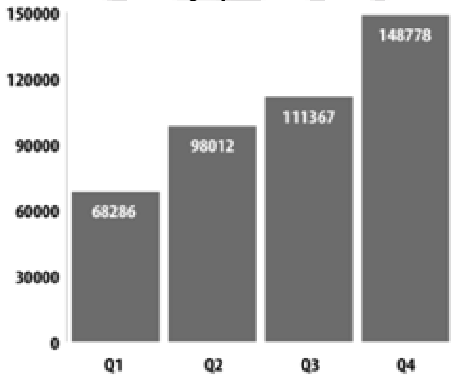


Fig. 2: Number mobile malware samples in our collection
Number of mobile malware samples in our collection
Among mobile malware, SMS Trojans are still leading the field is explained in figure3:

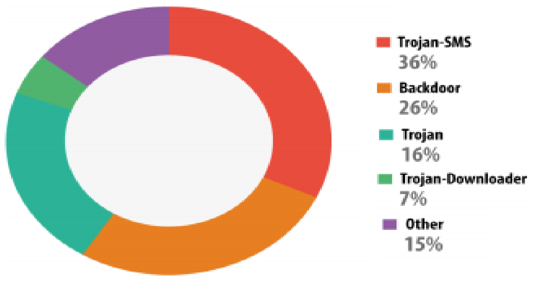


Fig. 3: malware distribution by behavior type

II. Related Work

In the following section, we give a short introduction to the field of malware detection research, and an overview on static and dynamic malware detection techniques.

A. *Malware Detection*

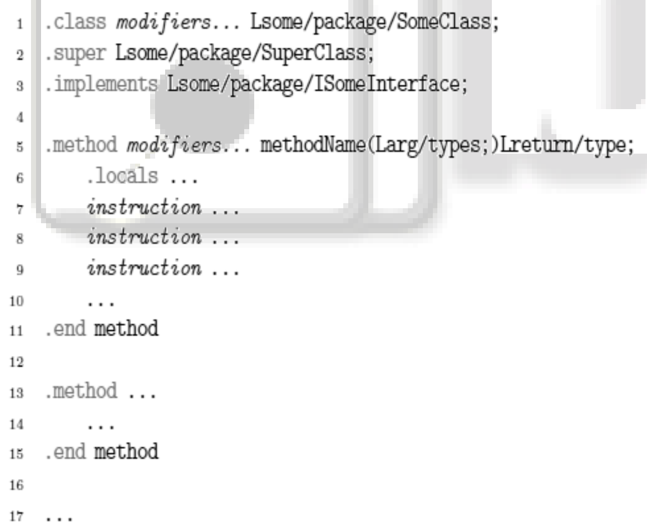
Generally malwares are detected during the execution of a particular application. A highly damage causing malware can affect the CPU usage, system performance etc. Related work is done based on hybrid model that is static and dynamic analysis were performed on the cloud[5].

AD Sandbox is an analysis system for malicious websites that focus on detecting attacks through JavaScript. JavaScript does not have a built-in sandbox. It thus provide an isolated environment for executing the websites [1].The outcome of the analysis is stored in database that contains information about the site is malicious or not [1].

Behaviour based analysis is performed based on algorithms detects suspicious applications [2]. The algorithms used to perform this analysis were the Naive Bayes algorithm, the Decision Tree algorithms and the Logistic Regression algorithm [2].

Android applications are written in java language and are executed in DVM (Dalvik Virtual Machine).DVM is optimized for the characteristics of android operating systems. DVM is used to translate java. Class files into dex formatted files [5]. .dex files are converted into smalifiles using backsmali [3].

The structure of such file is:



```
.class modifiers... Lsome/package/SomeClass;
.super Lsome/package/SuperClass;
.implements Lsome/package/ISomeInterface;

.method modifiers... methodName(Larg/types;)Lreturn/type;
    .locals ...
    instruction ...
    instruction ...
    instruction ...
    ...
.end method

.method ...
    ...
.end method

...
```

B. *Analysis Types*

The hybrid model performs two types of analysis. Static analysis generates report that the implemented application contains the malware or not. In case dynamic analysis generate a detailed report on the affected malware that contains logcat messages and kernel messages.

III. Design of the System

We now present an overview over the system together with some details about the design of the analysis systems.

A. *System Overview*

1) *Android SDK*

The Android software development kit supports most of the Java SE except for the AWT and Swing UI components. SDK is an emulator to run, debug and test end-user developed applications. The emulator is similar to a real device except some limitations regarding camera and video capture, headphones, battery simulation and Bluetooth. The emulator enables several operating systems to be executed on one machine and under different architectures. The emulator runs an Android Linux version on an ARM3 simulated processor.

2) *APK tool:*

Android Application Package provides to control a Android device from outside of Android system. It is used to find the malicious patterns generated in the small files.

3) *Android Debug Bridge (adb):*

Tool to enable communication with an emulator instance. It can be used to install applications to the emulator and transfer files to or from the emulator. It is also possible to issue command-line options to the operating system through a shell interface.

4) *Log cat message:*

provides a mechanism for collecting and viewing all logs issued within the emulator by the Android system and applications.

5) *Kernel Message:*

it generates the messages related to kernel. It contains the memory usage details.

6) *The Android Virtual Device (AVD):*

It is an emulator configuration that enables modelling of an actual device by defining hardware and software options that are then emulated. These options include: mapping to a system image, hardware features and dedicated storage area for simulating a SD card that contain user data. The system image contains the version-specific Android implementation that include the application framework and DVM.

IV. Static Analysis

Static analysis is a light weight mechanism for detecting malwares in smart phones [5]. The common approach is filtering binaries by malicious patterns called signatures.

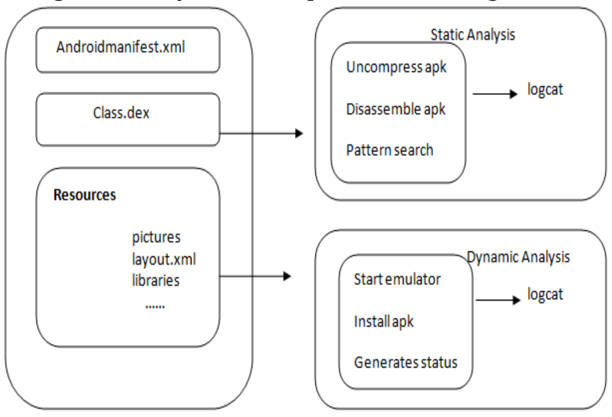


Fig. 4: Static Analysis and Dynamic Analysis

The analysis part is divided into two categories. The static analysis examines the code of the application to determine the properties of dynamic execution without running them [4]. Static analysis includes techniques such as decompression, decompilation and pattern searches shown in figure4. Android application sandbox takes .apk files as the input for the analysis. The APK is together packed with application resources which may contains pictures layouts and libraries, class.dex files, android manifest files as in figure5.An apk file is normally compressed zip file.

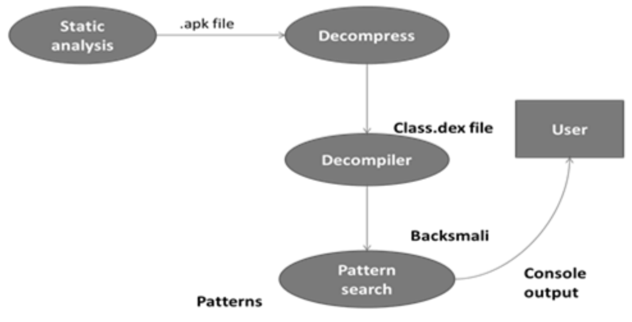


Fig. 5: Static Analysis

Decompression: is the first stage of static analysis. The uncompressed apk file contains

- Androidmanifest.xml
- Class.dex
- Resources

Android.xml is an XML file which holds the security permissions of the application. Class.dex contains the bytecode of the application which is to be interpreted by Dalvik Virtual Machine. Resources contain pictures, layouts and libraries.

Decompilation: The class.dex file holds the bytecode in the application. The class.dex files are decompiled into smali files by using backsmali. Smali files are human readable.

Pattern Search: The final step performed in static analysis. The disassembled code undergoes scanning for malicious patterns. The patterns are stored in database. As the android malwares are newly found day by day the server side contains admin, where he can update the patterns.

V. Dynamic Analysis

Dynamic analysis executes the application in a fully isolated environment. It provides low-level interaction with the system. Dynamic analysis is explained in figure6. It is more complex than static analysis. In dynamic analysis, the application is installed in android emulator from SDK bundle. After installation it performs dynamic analysis using APK tool. It automatically generates logcat messages and kernel messages. Kernel messages contain complete system state and hidden malicious activities.

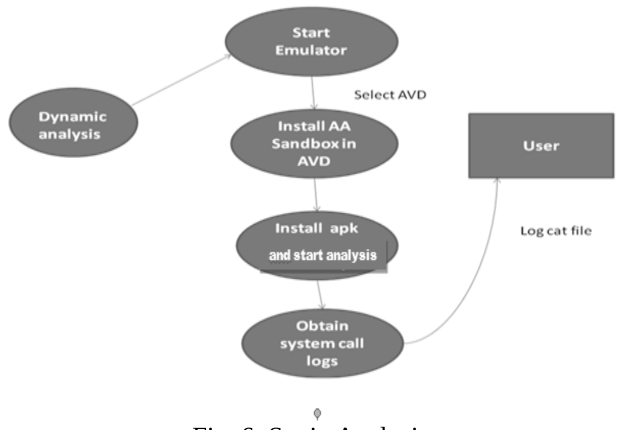


Fig. 6: Static Analysis

Dynamic analysis includes following steps:

A. *Prepare and start emulator:*

The android emulator creates a virtual phone's environment on desktop of the computer. Emulator supports AVD (Android Virtual Device) configurations. Emulator contains all software and hardware features of a smart phone except phone calls.

B. *Install AASandbox:*

The Loadable Kernel Module (LKM) assures creating a sandbox environment. The insertion of LKM into kernel of the emulator is done with the help of Android Debugging Bridge (ADB).Once LKM is loaded the output will be sent to logfile.

C. *Install APK:*

The APK is installed by using ADB. The installed APK will be unzipped and copied to directories. After installation the application is imported to android system and can be started manually.

D. *Obtain system call logs:*

When APK tool is finished execution the system calls will be generated.

VI. Conclusion

In this work we presented hybrid model for detecting malwares in android platform. This system is applicable as desktop service. The hybrid model presented here static and dynamic analysis. The hybrid model provides pre-check mechanism for malwares without installing the application. Whereas the antivirus check provides a mechanism after installing the application. By performing both static and dynamic analysis, the hybrid model provides a detailed study of the affected malware. The antivirus program cannot be able to generate the status of system performance when it is affected by malware.

References

[1] Andreas Dewald, Thorsten Holz, Felix C. Freiling. ADSandbox: Sandboxing JavaScript to fight Malicious Websites. SAC'10 March 22-26, 2010, Sierre, Switzerland.

[2] Abela, Kevin Joshua L, Angeles, Don Kristopher E, Tolentino, Robert Joseph. An Automated Malware Detection System for Android using Behavior-based Analysis. International Journal of Cyber-Security and Digital Forensics (IJCSDF) 2(2): 1-11 The Society of Digital Information and Wireless Communications, 2013 (ISSN: 2305-0012)

[3] Patrik Lantz. An Android Application Sandbox for Dynamic Analysis. Master's Thesis at Department of Electrical and Information Technology 1 November, 2011

[4] Michael Spreitzenbarth, Florian Echtler, Johannes Hoffmann. Mobile-Sandbox: Having a Deeper Look into Android Applications. SAC'13 March 18-22, 2013, Coimbra, Portugal.

[5] Erik Ramsgaad Wognsen, Henrik Sondberg Karlsen. Static Analysis of Dalvik Bytecode and Reflection in Android. Aalborg University, Master's Thesis, 2012.

[6] Eingereicht von Diplom-Informatiker Aubrey-Derrick Schmidt. Detection of Smartphone Malware. Tag der wissenschaftlichen Aussprache Berlin 2011
[7] http://media.kaspersky.com/pdf/KSB_2013_EN.pdf