Behavior Analysis of Android malware detection for Smart phone

Varsha Saxena[1], Dr. Sanjay Shrivastava[2], Shashikant Mourya[2]

[1]PG Student, [2]HOD, [3]Assistant Professor
Department of Computer Science, MGM COET, Plot No. A-09, Sector 62, Noida (NCR), Uttar Pradesh 201301

***Abstract*— *Smartphones are becoming more popular these days. Android operating system is dominating the smartphone market with 85% market share. The growth in Android based smartphones is encouraging malware authors to move various mobile app stores with malicious applications. This is done for unauthorized access of useful/private information stored in a smartphone by utilizing vulnerabilities in applications. This paper is intended to yield exhaustive literature survey and analysis of malware detection techniques on Android. To have effective detection techniques we have focused on the specific family i.e. AnserverBot family of Android malware, which is one of the largest Android malware family. We designed a tool based on specific features of AnserverBot family. These perticular features are collected via static and manual analysis of AnserverBot family. Our tool is capable of capturing all the malwares of AnserverBot family from a large collection of the applications. This detection scheme effectively detects the AnserverBot malware with high accuracy.*

Keywords- Android smartphones, Malware detection, smartphones operating systems, AnserverBot, smartphones applications.

I. INTRODUCTION

Android yields access to a wide range of useful libraries and tools that can be used to build rich applications. For example, Android enables developers to grab the location of the device, and allows devices to communicate with one another sanctioning rich peer–to–peer social applications. In addition, Android contain a full set of tools that have been built from the ground up aboard the platform providing developers with high productivity and deep understanding into their applications.

Android is a mobile operating system based on the Linux kernel developed by Google and designed primarily for touch screen mobile devices such as tablets and smartphones. Android's user interface is mainly based on direct manipulation,, such as tapping, swiping and pinching, to manipulate on-screen objects, using touch gestures that loosely correspond to real-world actions along with a virtual keyboard for text input. In addition to touch screen devices, Google has further developed Android Auto for cars, Android TV for televisions and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on game consoles, digital cameras, notebooks, and other electronics. The goals and contributions of this paper are threefold. First and second folds are included in static analysis. And third fold are in dynamic analysis.

Static analysis relies on features extracted without executing code, while dynamic analysis extracts features based on code execution (or emulation). In general, static analysis is more efficient, while static analysis is frequently more informative, particularly in cases of highly obfuscated code. Static analysis of an Android application can rely on features extracted from the manifest file or the Java bytecode, while dynamic analysis of Android applications can deal with features involving dynamic code loading and system calls that are gathered while the application is running. In this research, we analyzed the effectiveness of combining static and dynamic features for detecting Android malware.

First phase Involves two different bloom filters that classify a given sample into malware or benign class based on permission feature set only.

In second phase, the evaded malicious samples from Phase 1 are further analyzed by phase 2 consisting naïve bays classifier using permission and code based mixed features set.

"Inclusion of phase1 classification makes the technique computationally less intensive: while addition of phase2 classification improves the overall accuracy of the existing model"

Third phase involves the behavior analysis of android malware in which we detect the uncommon behavior of system calls by sandbox. The goal of sandboxing is to upgrade security by isolating an application to obstruct outside malware.

A sandbox is a security mechanism technique for unshared running programs. It is frequently used to execute untested or untrusted programs or code, possibly from unverified or untrusted third parties, users, suppliers or websites, without risking damage to the host machine or operating system.[1] A sandbox typically provides a closely controlled set of resources for guest programs to run in, such as rubbed space on disk and memory. Network access, the ability to examine the host system or read from input devices are usually prohibited or heavily restricted. Sandboxing is often used to test unverified programs that may include a virus or other malicious code, without allowing the software to damage the host device. In the sense of providing a highly controlled environment, sandboxes may be seen as a specific example of virtualization.

In this system we use a specific family of virus for detection. The specific family of virus is "AnserverBot" which is the most cosmopolitan and one of the largest Android malware families which influenced Android devices. The variants of this malware family contain licit apps with malicious code. Generally the malicious code communicates with remote Command and Control (C & C) servers for commands to berun on infected device. A large number of licit apps are being splendiferous with this malware family. These insidious apps are being distributed in various third party Android markets. The main characteristics of this family are that it has two invisible payloads. We will call these two payloads as payload A, payload B [22].

We used of a specific family of Android malware due to following reasons:

1. AnserverBot is the most sophisticated, well-informed Android malware family which effectively uses encryption and other techniques to escape itself from being detected.
2. We can manually analyze a specific family for its specific behavior, so that we can represent almost all malwares of that specific family.
3. Solutions are easier because of concentration on a specific family.

1.1 Android Malware

Mobile malware is malicious software that is specifically built to attack mobile phone or smartphone systems. These types of malware rely on exploits of particular operating systems (OS) and mobile phone software technology, and capture a significant portion of malware attacks in today's computing world, where mobile phones are increasingly common.

Mobile malware is malicious software that targets mobile phones or wireless-sanctioned Personal digital assistants (PDA), by causing the collapse of the system and loss or leakage of confidential information.

1.2 Malware are classified into five types according to their attack's ways in Android

1 Mobile Bot nets: Botnet is a remotely controlled malware which accept command from remote Command and Control (C & C) server. It permits taking complete control over the infected Android.

2 Ransom ware: Ransom ware is type of malware which abstracts the information of infected phone and ask for money to retrieve it back.

3 SMS Sending: This is a very famous Malware type in the Android as well as other smartphones. This type of malware is utilized to send SMS to premium numbers so the victim harm financially.

4 Spyware: Spyware is a malware that constantly monitor the Android and collect the sensible information about the user like phone-book numbers, MMS and SMS. It may also keep calls of the Android and transfer the voice data to the remote server without knowledge of the user.

5 Destructive Trojans: This type of malware, which mislead the user as it is licit app, are downloaded from the App store and may update or delete contents on the devices.

1.3 Malware families in Android

Malware family is a set of different malware having various signature and same functionality. By using some anxiety techniques we can develop different of the malware which have same malicious function as its root malware but its signature will be updated.

Some malware families of Android-

1 Ginger Master: It was first established by researchers from North Carolina State University in August 2011, Discovered in China. Android Ginger Master is a trojanized application family that targets Android mobile devices. Now there are about 6,122 known variants of this family [2].

2 DroidKungFu: It was discovered in 2011. DroidKungFu malware family is launched embedded in applications and efforts to get root access of the victim's device. Security developers said the malware seems to have back-door functionality that enables a stoner to install a malicious, run programs or navigate to a specific website, Android package.

3 JIFake: The fake mobile app has an embedded SMS Trojan functionality to transfer text messages to premium-rate phone numbers. JIFake malware family masquerades as an open-source instant messaging client for the ICQ network, mobile app for JIMM.

4 Basebridge: The malware, which composed 2% of the total Android malware threats, the Basebridge malware family acts as spyware, capturing sensitive information from the device owner to a remote location.

5 AnserverBot: It makes the influenced device as a bot driven by some C & C servers. The AnserverBot malware family piggybacks on legitimate apps.

1.4 Literature Review

Malware detection on smart phones are separated into static and dynamic types. In term of static detection, some features are removed from an app and analyzed before the app is executed. In term of dynamic detection, the app is executed in simulator and considered according to the log files. Both types have advantages and disadvantages.

1.4.1 Static Analysis

This technique is carried out without running the application in an Android emulator or device. Static analysis is a technique to observe malicious behavior by analyzing the code segments. However, this technique has a leading drawback of code obfuscation and dynamic code loading. The advantages of static analysis are that less time consuming, the cost of computation is down and low resource consumption. However, code obfuscation design the pattern matching a leading drawback in observing the malicious behavior.

X. Liu et al. Separated static analysis into two phases. Phase I utilized the permissions declared by the application for frequent analysis. Phase II is introduced if the application is known as malware in Phase I. Using a vector space model for classification contributes to efficiently observing malware make believe to be normal programs .The function call in the program code is analyzed in Phase II, so as to find out the original permissions utilized by the application. These will be designed by the permission accumulation of the given malware into a malware database.

Quan et al. predicted that mobile phone hardware restraints result in low detection rates and current behavior feature analysis methods are too difficult. Therefore, their methods develop sensible behavior feature vectors which corresponds multi-source behavior features from the three sensible behavior features utilized by the application: system call, API, dynamic execution of native program code. This vector can automatically represent down-level system behavior and high-level application malware behavior patterns.

1.4.2 Dynamic analysis

According to P.V.Shinjo and A.Salim ,Dynamic analysis is finished for abstracting the API calls designed by a binary file while in execution. Cuckoo is utilized to run and analyze malware files and create analysis result of the behavior of malware while in execution. In this experiment the cuckoo malware analyzer installed under Ubuntu 10.04 with VMware22 virtual machine is utilized as the safe domain. The record file obtains API calls designed during registry updation, execution, and the data such as heap memory address and process address.

The attackers utilize the same set of API to do malicious activities. So the presence or absence of an API in the record is not sufficient to indicting if the given file is malware or not. In our work we notice the API call sequence .APIs are yielded by the operating system to access the low level hardware by system calls for the application programs.

The correspondence in the call sequence between files in the corresponding class must be higher than the similarity between the files in the various classes. We utilize n-gram based method to analyze the call sequence called API-call grams. As the size of the n-gram grows the number of corresponding n-grams between two files inside the same class itself is very low. On the other hand, the analysis based on unigram is same as examining if the API is existing or not in a file. So in our work we observe only 3-API-call-grams and 4-API-call- rams. The feature vector is generated as follows. The set of three and four API-call-grams are created for each file from the processed call sequence record. For each n-gram set are grouped and grams below a limit is excluded. A table for both API-call-grams (3-API-call-grams and 4-API-call-grams) are generated in which the entries are API-call-grams from the n-gram set similar to a binary file in the dataset. Thus the table obtains a global list of API callgrams which in turn classified with ratio and we exclude some API-call-grams with low ratio. The selected API-call-grams represent the features.

1.5 System Framework

There are two parts of the proposed approach, static analysis and dynamic analysis. Static analysis is the traditional way of detecting the malware is based on the database system which keeps the hash of known malwares. When any suspicious binary is analyzed its hash is calculated and examined in the database if it exists or not. Whether it exists, then it is declared as a malware otherwise it is declared as benign software.

As different obfuscation techniques can escape this detection method, so we need advanced malware detection mechanisms which are not based on signatures fully. Although Static techniques are much usable in malware detection on Android but after applying the static analysis on smartphones, suspicious application is identified so there are reasons due to which dynamic analysis is preferable.

1.5.1 Static Analysis

This system implements static analysis in which testing cycle is divided in two phase.

Phase1-

First phase utilize two bloom filters: Bloom filter Malicious (BFM) and Bloom filter Benign (BFB).

These two bloom filters classified the collected malware samples into malicious and benign classes via permission based features. In this process the decision module also be used for decide which sample should be sent to next phase. Decision module decides the sample is belonging to malicious or benign class.

If the collected samples are declared benign by both filters, that sample belong to benign class

And if collected malware samples are declared malicious by both filters, that samples belong to malicious class and if samples are declared benign by one filter and declared malicious by other filter then it is called the unclassified sample. So it can say that which sample does not have same result in both filters that is called the unclassified sample. These unclassified samples sent to next phase.

Phase2-

Second phase uses the Naïve Bayes classifier. It is faster than other classification technique. It is so simple probabilistic classifier based on Bayes theorem with the assumption of independence between features.

In this, the evaded malware samples from phase 1, are further analyzed in phase 2 with naïve Bayes classifier. In phase 2, code based features module are used with permission based features modules and need a mixed based permission and code based features module which can mix the both features samples.

1.6 Dynamic Analysis

Dynamic analysis is the analysis technique of any software or application that is appeared after actual executing the programs. The behavior of the program is saved by executing it in a decent domain. We describe following methods of static analysis in our literature survey:

1.6.1 TaintDroid

However, it focused only on those apps that are shared to get sensible information. TaintDroid tracked the sensible information by labeling data in the memory and detected the privacy leaking in applications.

1.6.2 Andromaly

It noticed different acts created from the mobile device and then applies Machine Learning anomaly researchers to classify the collected data as benign or malicious. But it created various false positives. This is a host-based frame work which works on Android mobile devices for detecting malwares.

1.6.3 CrowDroid

In this paper developer did dynamic analysis of application behavior for observing malware in the Android platform. Developer gathered data set from a crowd of users and detects their behavior by logged system calls. It describe that by logged system call we can examine the behavior of the user more accurately. It did not focus on specific families in which we can take benefit of some Specific behavior of that family so that observe accuracy can be increased.

1.6.4 SCSDroid

It follows the concept that even if MRAs can be invisible as benign applications, their malicious behavior would still seem in the system calls. However, it detects many families of Android malware but it did not take some special behavior of respective families, so that there is a opportunity to grow detection correctness of specific malware families. SCSDroid is a mechanism to detect Malicious Repackaged Applications (MRAs) utilize thread grained system call sequences generated by applications.

II. TOOLS AND FRAMEWORKS

We describe various tools and framework in our literature surveys which are used to manipulate code of applications.

2.1 Androguard

Androguard is a tool written in python to play with: Dex (Dalvik virtual machine) _les, APK (Android application),Android's binary xml(.xml),. Its author is Anthony Desnos. Androguard has following features:

1 Manage DEX/APK format into full python objects.

2 Analysis a crowd of Android apps.

3 Reverse engineering of applications.

4 Modify Android's binary xml in classic xml(human readable)

2.2 ADAM

ADAM share several modification and Obfuscation techniques to create different variants of Android applications, and inspects the strength of various Android anti-malware systems. ADAM is a tool which automatically inspects the performance of anti-malware system.

2.3 Android SDK

The officially subsidized Integrated development environment (IDE) is Eclipse utilizing the Android development kit (ADT) Plugin. The Android Software development kit (SDK) is a group of development tools comprising a handset emulator based on QEMU(machine emulator and virtualizes),a debugger, libraries,. Computers running Linux, Windows XP or later supports Android app development process.

2.4 Virus Total

It gives the result of each famous antivirus product like Avira Antivirus, Antiy Antivirus , AVG Antivirus etc. for the application we want to analyze. Virus Total is a free online service that yields free checking of _les for botnets, malwares and other types of malicious contents. It uses about 52 variants of antivirus products and scan engines to inspect for malwares.

2.5 Strace

It acts Process-id as input and scan system calls used by that specific process during the run time. “Strace" is a debugging utility for Linux and some other Unix-like systems to scan the system calls utilized by a program and all the signals it takes. We used this tool to trace the system calls created by the Android application.

2.6 Monkey

“Monkey" tool provides an API to write programs. These programs are utilized to manage the Android device or emulator from outside of Android code. Using this tool we can install and run Android application automatically. We can also transfer keystrokes to Android application dynamically. It receives the package name of Android application as input and give random commands to that specific package. We used this tool to automatically run Android applications.

III. CONCLUSION & FUTURE WORK

As Android based smartphones are very popular these days, malwares targeting Android will increase. This dissertation is considered on a particular family of Android malware i.e. AnserverBot family, which is one of the largest Android family. We analyze particular and unique behaviors of AnserverBot family. We proposed a dynamic analysis technique that will found all the system calls with their parameters requested by the apps and analyze if it may be a AnserverBot malware or not. This technique has been applied on a data set of approximate700 apps which has 183 AnserverBot malware. We are able to catch all the AnserverBot malwares. Detection correctness of AnserverBot malware is 100%. An overall detection rate of 82.89% has been achieved. There are some false positives due to some similar behavior of other malware family. All the particular families can be detected with higher success rate by designing family specific solutions. So in future other families can be considered.

REFERENCES

[1] Wu Zhou Xuxian Jiang Yajin Zhou, Zhi Wang. Hey, you,get o_ of my market: Detecting malicious apps in o_cial and alternative android markets. IEEE/ACIS, 2012. vii, 6

[2] Machigar Ongtang William Enck and Patrick McDaniel. On lightweightmobile phone application certi_cation. CCS '09 Proceedings of the 16th ACMconference on Computer and communications security, pages 235{245, 2009.vii, 7

[3] Google. Virus total. https://www.virustotal.com/. viii, 10

[4] Uri Kanonov Asaf Shabtai. Andromaly: a behavioral malware detection framework. Springer Science+Business Media, 2011. vii, 8

[5] Google. Androguard. http://code.google.com/p/androguard/, 2011. viii,

[6] M. Ongtang, S. McLaughlin, W. Enck, and P. McDaniel,"Semantically Rich Application-Centric Security in Android,"in *Proceedings of the 25th Annual Computer Security Applications Conference*.

[7] Y. Yi *et al.*, "Dependency-based malware similarity comparisonmethod," *Journal of Software*, vol. 10, no. 22, pp. 2439-2446, 2011.

[8] E. Chien. Security response: Symbos.mabir, symantec, 2005.

[9] E. Chien. Security response: Symbos.skull, symantec, 2004.

[10] J. B. Guptil, Examining application components to reveal android malware, Retrieved on March 12,2016 from, https://www.google.com/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#

[11] Schmidt, A.D., Schmidt, H.G., Batyuk, L., Clausen, J.H., Camtepe, S.A., Albayrak,S.: Smartphone Malware Evolution Revisited: Android Next Target? In:Proceedings of the 4th International Conference on Malicious and Unwanted Software(MALWARE) (Oct 2009)

[12] Zhou, Y., Wang, Z., Zhou, W., & Jiang, X. (2012). Hey, you, get off of mymarket: Detecting malicious apps in official and alternative Android markets.*Proceedings of the 19th Annual Network and Distributed System SecuritySymposium*, 5-8 February 2012, San Diego, California, USA.

[13] A. Miklas, K. Gollu, K. Chan, S. Saroiu, K. Gummadi, and E. de Lara. Exploiting social interactions inmobile systems. In *UbiComp*, 2007.

[14] A. Mtibaa, A. Chaintreau, J. LeBrun, E. Oliver, A.-K. Pietilainen, and C. Diot. Are you moved by yoursocial network application? In *WOSN*, 2008.

[15] B. H. Bloom, "Spaceitime tradeoffs in hash coding with allowableerrors," *Commun.* ***ACM,*** vol. 13, no. 7, pp. 422-426, July 1970.