International Journal of Integrated Engineering:
Special Issue 2018: Data Information Engineering, Vol. 10 No. 6 (2018) p. 203-208.
© Penerbit UTHM DOI: https://doi.org/10.30880/ijie.2018.10.06.029

Understanding the Root of Attack in Android Malware

Rahiwan Nazar Romli[1*], Mohamad Fadli Zolkipli[1], Ahmad Al-Ma'arif[2], Muhamad Ramiza Ramli[1], Mohamad Aizi Salamat[3]

[1]Faculty of Computer System & Software Engineering, University of Malaysia Pahang, 26200 Gambang, MALAYSIA.
[2]School of Industrial Engineering, Telkom University, 40257 Bandung, West Java, Indonesia
[3]Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, 86400 Parit Raja, Batu Pahat, Johor, Malaysia.

Received 28 June 2018; accepted 5 August 2018, available online 24 August 2018

Abstract: With the rapid pace of development and change in mobile device technology and Android versions, Android malware has emerged and become a focus in current research. Subsequently, security and privacy have become one of the main issues in android malware. Therefore, it is essential to understand the behavior of Android malware in order to conceive an effective technique in malware detection and analysis. This article presents a comprehensive study regarding Android platform, its features in android malware code and also discusses the result from previous studies in order to support forward-looking in Android study.

Keywords: Android, Malware, Malware Detection and Analysis

1. Introduction

The rise of cheaper mobile devices and internet connection has become a contributing factor to the increase of mobile user. Among all, Android is the most popular and also known as the most wanted OS by user [1][2][3]. Due to this development, the security of Android OS becomes a major challenge. With the device always connected to internet, it becomes vulnerable to malware attack through the applications installed in the device. A report by [4] concluded that over 100 billion malwares were detected, and from that, nearly100 million were in Android platform. In an attempt to evaluate the progress of these studies within specific areas, this article is sectioned as the following:

- The first section presents Android overview and its architecture. It describes about Android operating system and its importance in order to provide better understanding about mobile malware. Most of the previous studies focus on malware techniques and have ignored the basic part in malware analysis.
- The second section presents an inclusive study about the type of malware in Android environment. This part presents a brief review about top malwares in android, the way they attack, the after-effect of the attack and the medium used to inject these malwares in android platform.
- The third section presents the comparative analysis on existing work. Similar studies are reviewed in order to identify the methods used in malware analysis; the techniques applied in malware detection; and also the advantages and disadvantages of each. This section is meant to provide insight into the progress of this study in the areas mentioned.

2. Android Overview

For a simple definition, Android is a mobile operating system for mobile phones and tablets that is open source. Android was introduced by Google in 2008. It was designed based on Linux Kernel.

The long written history of the working framework for portable Android applications started with Android beta in November 5, 2007. Later on, Android 1.0 was released worldwide in September 2008.This version of Android was developed through the cooperation of Google and Open Handset Alliance. Since then, various updates to its base working framework have been seen.

The Android versions 1.0 and 1.1 were not released worldwide under unique names. Each Android version has been uniquely named after dessert and arranged in alphabetical sequence since 2009's Android 1.5 Cupcake. The most recent one is Android 8.1 Oreo that was released worldwide in December 2017. The following Table 1 demonstrates the historical background of Android forms.

Table 1 : Android Version history

Code Name	Android Version	Year Release	API Level	Security
NoCode Name	1.0	2008	1	Unsupported
Petit Four	1.1	2009	2	Unsupported
Cupcake	1.5	2009	3	Unsupported
Donut	1.6	2009	4	Unsupported
Éclair	2.0 - 2.1	2009	5-7	Unsupported

*Corresponding author: rahiwan@ump.edu.my

Froyo	2.2 - 2.2.3	2010	8	Unsupported
Gingerbread	2.3 - 2.37	2010	9-10	Unsupported
Honeycomb	3.0 – 3.2.6	2011	11-13	Unsupported
Ice Cream Sandwich	4.0 – 4.0.4	2011	14-15	Unsupported
Jelly Bean	4.1 - 4.3.1	2012	16 -18	Unsupported
KitKat	4.4 – 4.4.4	2013	19 -20	Unsupported
Lollipop	5.0 - 5.1.1	2014	21 -22	Supported
Marshmallow	6.0 – 6.0.1	2015	23	Supported
Nougat	7.0 – 7.1.2	2016	25 – 25	Supported
Oreo	8.0 – 8.1	2017	26 -27	Supported

2.1 Android Architecture

Android operating system's basic design is based on Linux kernel and it runs all Java written applications in isolation [5][6]. There are two principles in android, one being, Android could not kill the running application if user switch to other application at the same time, another one is, Android will kill applications when the memory usage is high but it will save the app state for when the phone restarts at other time. Android architecture as shown in Fig. 1 can be classified into four layers which are Application, Application Framework, Libraries + Runtime and Linux Kernel.

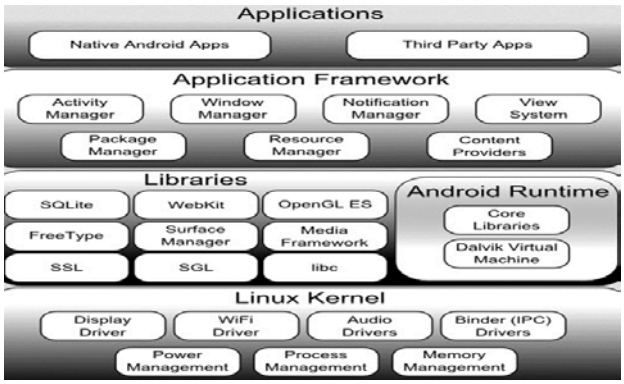

Fig. 1 Android Architecture

Each layer in Android has different task thus provides consistence in the service. Application layer is developed with the help of APIs from core libraries and Android framework. Most mobile device contains basic applications such as Calendar, Contacts, SMS, browser and other applications that can be downloaded from android store.

Application Framework supply the more elevated amount of administration to Application. This layer gives abnormal state APIs to android application empowering them to execute custom highlights. On the highest point of Linux Kernel is an arrangement of libraries. These libraries are utilized to empower different highlights in the Android OS. All these are composed in C++. Inside this layer comes the most essential part in Android Architecture which is Android Runtime. Android Runtime essentially has two parts which are Dalvik Virtual Machine (DVM) and Core Libraries.

Dalvik Virtual Machine is considered as the most critical segment in Android OS design. Everything over this level of the design is composed in Java. Thus, essentially it contains an arrangement of .class records. Running those .class records on a little portable processor however, is an issue. Hence, Dalvik Virtual machine changes these .class records into .dex documents which would make preparing substantially quicker. These records keep running with least memory impression. Presently it has numerous occasions of this VM running i.e. performing various tasks. Dalvik Virtual Machine is like JVM except it is outlined and streamlined for Android stage.

The base of the Android design is Linux. This part is focal module of an OS. It is in charge of memory administration, process administration, and plate administration. Essentially it associates the framework equipment to the application programming.

Android needs a part and instead of composing its own particular, they pick Linux. As Linux is open source operating system, Android developers could alter the Linux part to fit their needs. Linux gives the Android designers a pre-assembled version; officially kept up working framework portion to begin with. This is the way a wide range of gadgets are assembled.

2.2 Android Security Issue

Android security model is designed in multilayer in order to provide protection for android user. The adaptability of the stage permits designers of all experience levels to effortlessly work with the SDK to assemble secure applications. There are protected APIs placed between Application and Libraries. All of the Android programs must be given permission by Android Permission System when accessing a particular application.

Permission is the right that a particular application has that enables it to play out specific activities on their device [7]. This Permission is defined in Manifest file AndroidManifest.xml, which is compulsory for shipping each android app.

For example, when a user uses Camera to perform action of taking picture, the Android system will check whether the application file has the CAMERA permission. Previous studies regarding the inefficiency of Permission can be found in[8][9][10][11]. Unfortunately, as this permission also allows anti-virus application, it could allow malware author to inject their malicious into Android system.

However, this Permission has a few flaws. A client cannot choose to allow single permission, while denying others. Numerous users, in spite of the fact that an application may ask for a suspicious permission among many apparently genuine consents, will even now affirm the installation. This is called as all-or-none policy.

On the other hand, most of the time, user cannot pass judgment on the suitability of permission for the application being referred to. At times it might be self-evident, for instance when a diversion application asks for the benefit to reboot the device or to send instant messages. By and large, in any case, user will usually be unequipped for surveying permission propriety.

Lastly is about circumvention in permission. Worth, which should be executable just given the proper permission, can at present be gotten to with less permission or even with none by any stretch of the imagination.

Based on aforementioned details, the security in Android framework can be affected by malware through abusing of permission feature in Android. However, permission is the most conspicuous element that has been executed by many researchers[12][13][14][15],as shown in Table 1, the ranking of permission feature from year 2010 – 2016. Currently, the number of android malware using INTERNET permission as full internet access remains highest since 2010. With the update of android versions and the growing of malware, there are a few rising of permission feature between year 2016 and 2017 such as Bluetooth and Receive_Boot_Complete.

Table 2 Ranking of Permission feature between 2010-2016

2010	2012	2013
1.Internet	1. Internet	1. Internet
2.Read_Phone_state	2. Access_Network_State	2.Access_Network_State
3.Vibrate	3. Vibrate	3. Read_Phone_State
4.Write_External_Storage	4. Access_Fine_Location	4. Vibrate
5.Access_Network_State	5. Read_Phone_State	5. Access_Wifi_State
6.Send_SMS	6. Wake_Lock	6. Wake_Lock
7.Wake_Lock	7. Access_Wifi_State	7. Access_Fine_Location
8.Receive_Boot_Completed	8.Write_External_Storage	8. Write_External_Storage
9.Access_Wifi_State	9. Access_Coarse_Location	9. Factory_Test
10.Access_Fine_Location	10. Factory_Test	10. Access_Coarse_Location
2014	**2015**	**2016**
1. Internet	1.Internet	1. Internet
2. Access_Network_State	2.Write_Sync_Setting	2. Access_Wifi_State
3. Read_Phone_State	3.NFC	3. Modify_Phone_State
4. Vibrate	4.Read_History_Bookmarks	4. Read_Phone_State
5. Wake_Lock	5.Access_Coarse_Location	5. Access_Fine_Location
6. Access_Wifi_State	6.Location_Hardware	6. Access_Coarse_Location
7. Access_Fine_Location	7.Read_Call_Log	7. Wake_Lock
8. Write_External_Storage	8.Add_Voicemail	8. Vibrate
9. Factory_Test	9.Access_Wifi_State	9.Access_Network_State
10.Access_Coarse_Location	10.Access_Fine_Location	10. Restart_Package
	11. Read_SMS	11. Read_Contact
	12. Send_SMS	12. Read_Phone_State
		13. Camera
		14. Set_Wallpaper
		15. System_Alert_Window

Based on the comparison of previous works (as shown in Table 2), it can be concluded that there are some new features used by malware authors in creating malware. The new features include *READ_HISTORY_BOOKMARKS*, *CHANGE_WIFI_STATE* and *GET_TASKS*. Fig. 2 shows the overall android feature permission that is highly used by malware authors.

2.3 Comparative Study on Existing Work

A considerable measure of study was done in portable stage and distributed computing. Malware distinguishing proof and investigation were connected in this study. Here, we bring up the previous work by the other researcher.

Hanling Zhang presents ScanMe Mobile for malware investigation utilizing cloud stage [16]. The job of this model is to give clients point by point data about Android Application Package (APK) records before introducing them on their gadgets. With ScanMe Mobile, clients can transfer APK records from their gadget SD card, filter the APK in the malware identification framework that could be sent in the cloud, assemble a complete report, and store or offer the report by distributing it to the site[16].

Three-layer mixture framework with lightweight antimalware motor proposed by [17].This examination allows quick time in malware identification, shield client from malware and diminish the data transmission among customer and the cloud.

S. Zonous[18]proposed a structure named Seacloud.It was produced for Android stage. It was outlined for better security arrangement utilizing cloud based. Seacloud imitates an enrolled cell phone gadget inside an assigned cloud and keeps it synchronized by consistently passing the gadget data sources and system associations with the cloud. This enables Seacloud to play out an asset concentrated security examination on the copied imitation that would some way or another be infeasible to keep running on the gadget itself.

While Jianlin Xu[19] created a framework named MobSafe. The target of this framework is to distinguish and quantifythe portable application that is benevolent. The mix of two systems, static and dynamic examination

strategy to gauge time required in assessed all android application on one market stage. From the consequence of development, distributed computing and information mining will play a job as to check the application that free from malware.

Following the study by Osamah L Barakat[20]that acquainted new methodology with improve malware analyser execution in malware investigation. Utilizing distributed processing, it opens approach to crowd source for the administration henceforth reassuring malware revealing and quicken malware discovery by drawing in general clients.

Lastly, John Oberheide[21] develop a model made out of a Windows based host specialist and an in-cloud investigation benefit and assess it utilizing a various dataset of 5066 one of kind malware executables. By relating data between various location engines, our framework gives more than 98% identification inclusion of the malevolent executables utilizing eight antivirus engines and two conduct motors contrasted with a 54% to 86% discovery rate utilizing the most recent commercial antivirus items .

Based on Table 3, this study defines there are few problems are related to mobile malware analysis. There are:

1. Resource in mobile such as storage, memory and processer are limited. This is shown in previous research works in row 9 and 10 in Table 3.
2. Weakness in malware detection technique. This is shown in previous research works in row 2 ,3 and 4 in Table 3.
3. There are thousands feature in malware code and need to look the relevant feature in malware analysis. This is sample of feature is shown from previous work located at row 8 in Table 3.

Based on deduction done, this work will shift focus towards android feature permission since malware authors are focus more on these permissions.

3. Summary

All the objectives for this paper has been answered. in all written sections. Existing Android architecture and security issues has been discussed. Table 3 has shown the comparison of existing works.

As of late, the versatile distributed computing is turning into another hot innovation. What's more, the security answer for it has moved toward becoming an examination center. With the advancement of the portable cloud processing, new security issues will happen, which needs greater security approaches. In this study, we compactly checked on favorable circumstances and models of portable distributed computing, what's more, broke down security and protection issues from three layers, which are portable terminal, versatile system and versatile cloud. At that point, as indicated by the issues we gave the current methodologies, for example, hostile to malware, security assurance, key administration and encryption, get to control, etc. It is hope that the review of issue can a guideline to many authors in conducting their works such as the network issue provided by [22] that focus on device to device communication

References

[1] S. Corporation, “Internet Security Threat Report,” vol. 21, no. 7, p. e98790, 2016.

[2] Ericsson, “Mobility Report,” White Pap., no. May, pp. 7–8, 2016.

[3] M. Butler, “Android: Changing the mobile landscape,” IEEE Pervasive Comput., vol. 10, no. 1, pp. 4–7, 2011.

[4] MalwareBytes, “State of Malware Report,” p. 11, 2017.

[5] P. Singh, P. Tiwari, and S. Singh, “Analysis of Malicious Behavior of Android Apps,” Procedia Comput. Sci., vol. 79, pp. 215–220, 2016.

[6] S. Brähler, “Analysis of the Android Architecture,” Os.Ibds.Kit.Edu, p. 52, 2010.

[7] B. Chalise, “Android Permission,” 2015.

[8] K. Tam, A. L. I. Feizollah, N. O. R. B. Anuar, R. Salleh, and L. Cavallaro, “The Evolution of Android Malware and Android Analysis Techniques,” vol. 49, no. 4, pp. 1–41, 2017.

[9] A. Nayak and A. Pons, “Fuzzy Logic Based Android Malware Classification Approach,” no. June, pp. 1–8, 2014.

[10] G. Andrejková, A. Almarimi, and A. Mahmoud, “Approximate Pattern Matching using Fuzzy Logic ∗,” vol. 1003, pp. 52–57, 2013.

[11] M. Zheng, M. Sun, and J. C. S. Lui, “DroidAnalytics : A Signature Based Analytic System to Collect , Extract , Analyze and Associate Android Malware.”

[12] P. Wijesekera, A. Baokar, A. Hosseini, S. Egelman, D. Wagner, and K. Beznosov, “Android Permissions Remystified - A Field Study on Contextual Integrity,” 24th USENIX Secur. Symp. (USENIX Secur. 15), pp. 499--514, 2015.

[13] M. Frank, B. Dong, A. P. Felt, and D. Song, “Mining permission request patterns from Android and Facebook applications,” Proc. - IEEE Int. Conf. Data Mining, ICDM, pp. 870–875, 2012.

[14] H. Shahriar and M. Islam, “Android Malware Detection Using Permission Analysis,” 2017.

[15] N. Peiravian and X. Zhu, “Machine learning for Android malware detection using permission and

API calls," Proc. - Int. Conf. Tools with Artif. Intell. ICTAI, pp. 300–305, 2013.

[16] Y. Cole et al., "ScanMe Mobile: A Cloud-based Android Malware Analysis Service," SIGAPP Appl. Comput. Rev., vol. 16, no. 1, pp. 36–49, 2016.

[17] S. Alam, I. Sogukpinar, I. Traore, and Y. Coady, "In-cloud malware analysis and detection: State of the art," ACM Int. Conf. Proceeding Ser., vol. 2014–Septe, 2014.

[18] S. Zonouz, A. Houmansadra, R. Berthiera, N. Borisova, and W. Sanders, "Secloud: A cloud-based comprehensive and lightweight security solution for smartphones," Comput. Secur., vol. 37, pp. 215–227, 2013.

[19] J. Xu et al., "MobSafe: cloud computing based forensic analysis for massive mobile applications using data mining," Tsinghua Sci. Technol., vol. 18, no. 4, pp. 418–427, 2013.

[20] O. L. Barakat et al., "Malware analysis performance enhancement using cloud computing," J. Comput. Virol. Hacking Tech., vol. 10, no. 1, pp. 1–10, 2013.

[21] J. Oberheide, E. Cooke, and F. Jahanian, "Rethinking antivirus: executable analysis in the network cloud," Proc. 2nd USENIX Work. Hot Top. Secur., p. 5:1–5:5, 2007.

[22] M.S.M. Gismalla, M.F.L. Abdullah, "Device to Device Communication for Internet of Things Ecosystem: An overview", International Journal of Integrated Engineering. vol. 9, no. 4, pp. 118-123, 2017.

Table 3 Malware Analysis

Previous Study	Year	Platform	Method of Malware Analysis	Technique	Disadvantage
Cloud Based Malware Detection Technique	2017	Cloud	Detection		Implement the tool in cloud to detect malware
Scan Me	2016	Android + Cloud	Detection Static Analysis Classification	Artificial Neural Network	Not real time basis Using SD card to send APK file to cloud One-way communication
Pattern Matching Techniques for Metamorphic Virus Detection	2016		Detection	Pattern Matching	Apply to conventional method
Deep Learning for Classification Of Malware System Call Sequences	2015			Neural Network	Problem in malware classification
Mobsafe	2013	Android & Cloud	Static & Dynamic Analysis		No synchronization between mobile & cloud platform
Droidanalytics: A Signature Based Analytic System to Collect, Extract, Analyze and ASSOCIATE, ANDROID Malware	2013	Android	Detection Static analysis Dynamic analysis		
Profiling Mobile Malware Behaviour Through Hybrid Malware Analysis Approach	2013	Android	Hybrid		
Secloud: A Cloud-Based Comprehensive and Lightweight Security Solution for Smartphones	2013	Android & Cloud	Detection		limitation in few aspects 1.File system consistency 2.User privacy 3.Environment resiliency 4.Encryption
Analysis of Malicious and Benign Android Application, 2012	2012	Android	Detection Dynamic Analysis Classification		Limited of mobile resources Few malwarescan't beclassified

CROWDROID: A Framework for Behaviour-Based Malware Analysis In The Cloud	2011	Android	Detection Dynamic Analysis	K-Mean Clustering	Privacy Limited in mobile resources
Malware Behavioral Analysis System: Twman	2011		Dynamic Analysis		
A Framework for Behavior-Based Malware Analysis In The Cloud	2009	Cloud	Detection		
Virtualized In-Cloud Security Services for Mobile Device	2008	Other + Cloud	Detection		Running malware in cloud platform One-way communication