International Journal of Security and Its Applications
Vol. 9, No. 3 (2015), pp. 335-346
http://dx.doi.org/10.14257/ijsia.2015.9.3.26

The Analysis of Android Malware Behaviors

Fan Yuhui and Xu Ning

Department of Computer and Information Engineering, Huainan Normal University, Huainan, China
122336956@qq.com

Abstract

Currently the intelligent terminal based on the Android has occupied most of the market, and the number of malware aiming at Android platform is also increasing. The problems of security threats and privacy disclosure caused by malicious behaviors are becoming more serious. How to make the security assessments and metrics effectively for the security of application has become a research hotspot in recent years. In this paper, we use static behavioral analysis approach, the thesis analyzes Android malware, summarizes its malicious behaviors and its ways of stealing private data, and puts forward the methods of detection and prevention.

Key words: *Android application, malware, permission, decompilation, static detection*

1. Introduction

With the reduction of smartphone's cost and the rapid development of the application software based on smartphones, the smartphone becomes more and more popular in people's life in recent years. Due to the characteristic of openness, the share of Android smartphone is continually increasing in smartphone market. According to IDC, the market share of Android smartphone is up to 81 percent till the third quarter of 2013, which far exceeds all the other competitors.

Android smartphone has brought great convenience to people's life, at the same time, it causes problems of security. CNCERT has detected 702, 861 mobile internet malicious sample programs in 2013, among which 99.5 percent aims at Android platform [1]. These malicious programs not only affect smartphone users' normal use, but also have security threats such as malicious fee deduction, stealing information and remote control, which bring loss to smartphone users. In terms of intentions of these mobile malware, the malicious fee-deducting malware continues to take the first place (71.5%), fee consumption (15.1%) comes to the second place, and the followings are system damage and stealing information, accounting for 3.2%, as shown in Figure 1.

ISSN: 1738-9976 IJSIA

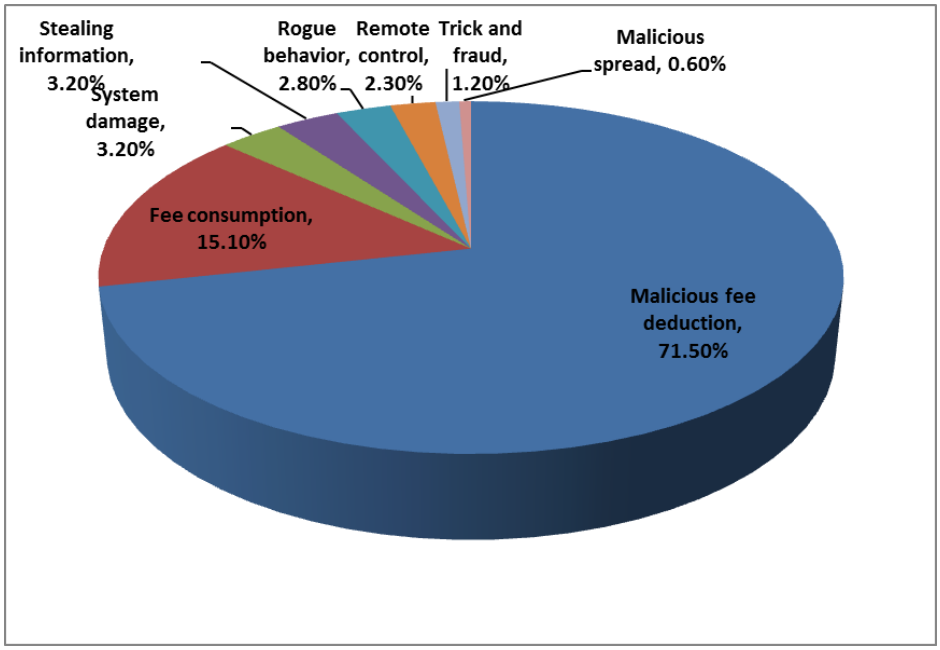

Figure 1. Intention-based Categories of the Mobile Malware in 2013

According to the first half-year report from NetQin in 2013, the malware of fee deduction deducts the users' fee through subscribing SP by ways of short messages, which causes the loss of 4.5 million to Chinese smartphone users each day; And the malware of remote control makes profits through accepting server commands to download the software in network and forcibly sells to smartphone users, and its profit is up to 7.8 million per day. So the detection and prevention of malicious software for smartphones has become an important issue for Network operators and the departments of cyber security to solve immediately.

2. Research Review

The experts and scholars at home and abroad have done some researches about the malware of smartphones and have gained some achievements. For instance, M. Miettinen and P. Halonen's essay analyzes the malware detection on mobile smart devices and indicates its problems and insufficiency [2]. The essay from Enck (Enck, *et al.*, 2010) focuses on the stealing privacy caused by the malware and comes up with the relevant scheme of monitoring [3]. Collin Mulliner and Aubrey-Derrick Schmidt from Technische University Berlin have also done some researches on smartphone malware [4~5]. Abhijit Bose's essay points out that we can use SVM to detect malicious behaviors of mobile handsets, and establishes the relevant detection model[6].

Wang Fei fei (Wang Fei Fei, 2012) find a detection method developed a signature_based malicious code in his paper [7]. Androguard [8], a famous Android malicious code detection tools, can detection the malware based on developer signature matching. These two methods can quickly and efficiently find the known malicious application, but unable to determine whether the new application has malicious behavior.

In order to effectively determine whether an unknown application for malicious applications, Enck (Enck, *et al.*, 2009) developed a lightweight application detection tool –Kirin [9], Kirin provides a solution for Android application detection which can be customized strategy. Another effective method according to the statistical data of the

application request permissions and call the API, using data mining technology to determine whether the target application contains malicious behavior. Yang Huan(Yang Huan, *et al.*, 2013) extract the request permission information for the Android application to construct the characteristics set, and used the permission sequential pattern mining algorithm could discover permission sequential pattern from 49 malware families and build the permissions association dataset to detect Android malware [10]. Peng (Peng H, *et al.*, 2012) using the similar idea, in their paper, they propose a method using probabilistic generative models for ranking risks of android applications [11], and obtain good results. Sangho Lee (Sangho Lee, *et al.*, 2013) introduce with analysis of the method to prevent an installation of malicious applications using permissions using Maximum Severity Rating (MSR) classification in their paper [12]. All these methods based on data mining can effectively detect unknown application if there have malicious behavior, but the Android applications request excessive permissions is ubiquitous, the data mining based on request permissions has too many false positives.

Kwang (Yan L K, *et al.*, 2012) developed a detecting system called DroidScope based on seamlessly reconstructing the OS and Dalvik semantic views for dynamic Android malware analysis, this system has three layers of hardware layer, the system layer and the Dalvik virtual machine layer to monitor the Android API [13]. The three layers system can facilitate the researchers defined analysis strategy. The researchers can use the interfaces of DroidScope provided to collect the code in local and the behavior in Java in order to achieve a variety of rich security strategy. Yang (Yang Z, *et al.*, 2013) think the sensitive data transmission without the participation of users is a very suspicious malicious behavior, they developed APPIntent framework can determine whether the sensitive data transmission was by user intention or not [14]. You Joung Ham (You Joung Ham, *et al.*, 2014) analyzd the normal system call event patterns from the most highly used game app in the Android open market, and the malicious system call event patterns from the malicious game apps extracted from 1260 malware samples distributed by Android MalGenome Project, then using the strace tool, system call events are aggregated from normal and malicious application [15].

At present, the study of smartphone malware has gained some achievements but the solving methods are mainly borrowed from techniques in computer platform. To sum up, the present study of smartphone malware is focused on the following aspects: 1. The study of malicious software including mobile botnets' attacks on smartphone users, Telecommunication Network and Internet and its resulting security threats; 2. The study of the design of mobile botnets based on the traditional botnets design, which includes mobile botnets' network architecture, its ways of transmission, its command and control on the net and its communication algorithm, *etc.*.

3. The Detection Analysis of Android Malicious Software

The Android system uses Linux as the kernel, and follows part of the Linux access control mechanism. The Android systems also use sandbox mechanism, permissions mechanism and the signature mechanism in order to protect the private of mobile phone user.

3.1. The Security Analysis of Android Platform

Android platform is developed by Google Company based on Linux2.6, which is composed of Linux and Java. It adopts the layered architecture design including Linux Kernel, Libraries and Android Runtime, Application Framework, and Applications, as shown in Figure 2.

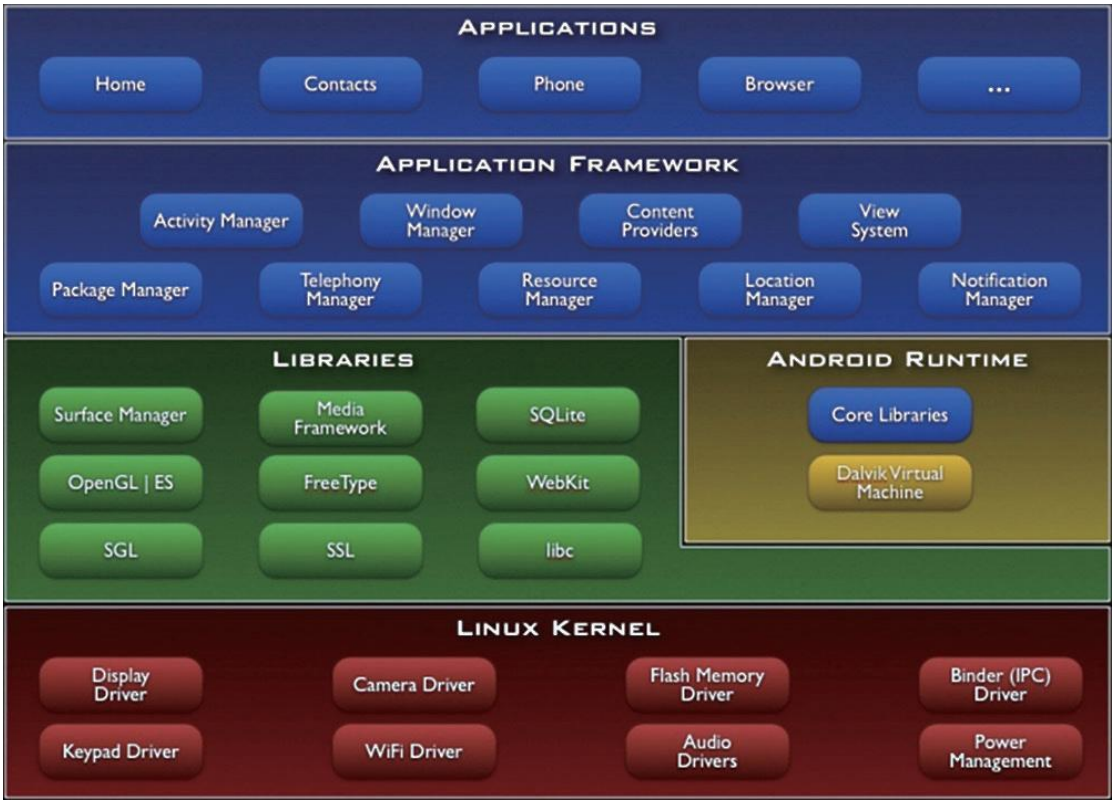

Figure 2. Android System Structure

3.1.1. Sandbox Mechanism: Using Java, Android applications operate on Dalvik VM, to which Android runtime environment provides Java core libraries. Each application runs with a unique system identity (Linux user ID and group ID). Each parts of the system were also using their own independent identification mode. When an Android application runs, it presents in the system as a single process and there is isolation between processes, as shown in Figure 3.

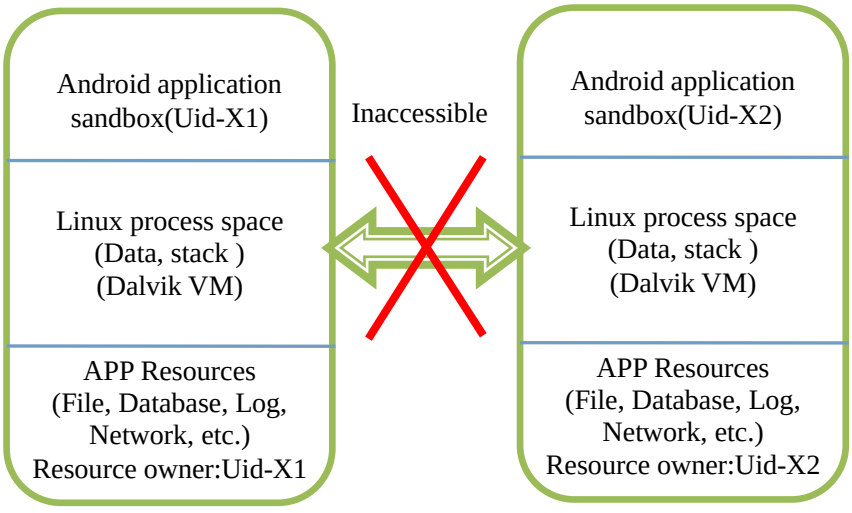

Figure 3. The Schematic of Application Sandbox

By using this mechanism, it can protect the independence during the application run. If the applications run in programs, system can close this Dalvik VM instance to protect the safe of the system.

3.1.2. Permissions Mechanism: Each Android application has an AndroidManifest.xml file which consists of the permission to run the application. If the application needs to use other permission which is provided by the AndroidManifest.xml file, it will be prevented or terminated by the system and the application will reminds the user of the needed permission before setup, as shown in Figure 4.

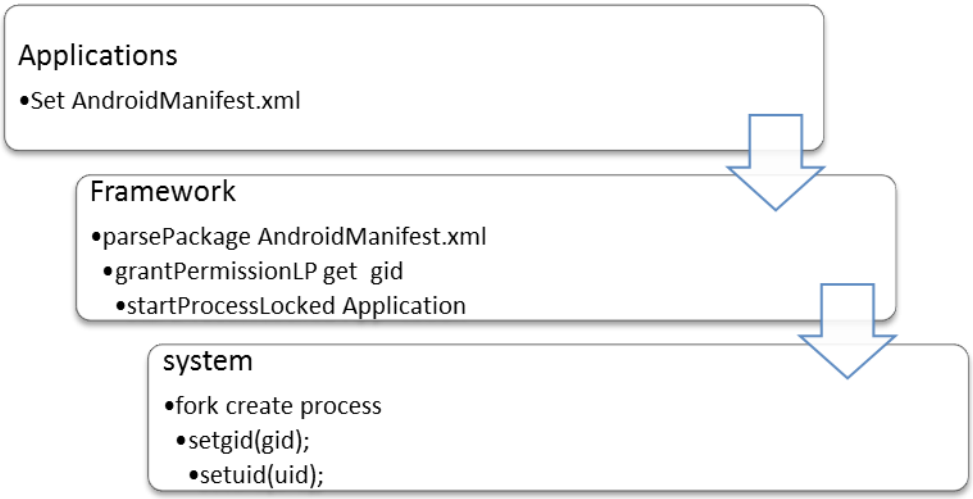

Figure 4. Android Application Permissions Flow Chart

As the security measure of Android system, it can control the application's behaviors which are beyond its limit, it cannot prevent the application to use the acquired permission to have malicious behaviors. To most users, they usually do not carefully check the access permission that the application applies for when installing it.

3.1.3. Signature Mechanism: Another security measure that Android system adopts is the file signature of APK applications. When releasing Android applications, APK applications can use Debug Key tool to compile and sign. This signature mechanism can protect the homology of the applications, so when a modified malicious application is installed on the Android system, the system will not allow this application to be installed or upgraded as the modified malicious application cannot match the original signature. The file signature system can only protect the installed applications not to be modified maliciously, but it cannot protect the newly installed applications and the pre-existing applications which already contain malicious behaviors, as shown in Figure 5.

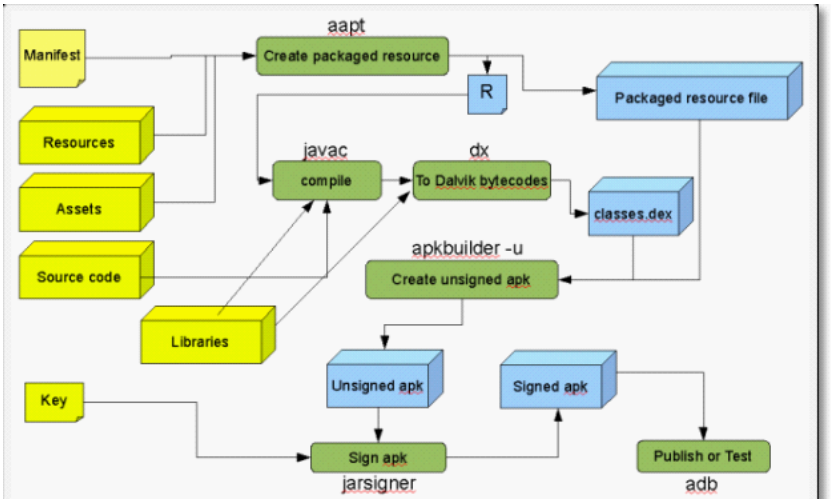

Figure 5. Android Sign Apk Process

When Android applications are published on Google Play platform, the developer's register account is needed, at the same time, the forthcoming applications must be tested by Google Play Developer Distribution Agreement (DDA) and Google Play Developer Program Policies (DPP). To the applications which violate the agreements and provisions, the publication will be suspended and the developer will be informed; to the malicious software, Google Play Store can unload user's software remotely. Google Play takes some measures to check Android applications, but Google Play Store is not like Apple App Store and Windows Phone Marketplace which check the forthcoming applications rigorously and only allow the qualified ones to be published. There are many Android online applications stores in different parts of the world publishing unchecked applications, which causes the wide spread of Android malware. Besides, Brush package and social networks provide more convenient ways to spread malicious software.

3.2. Detection Methods of Android Malware

Presently, there are two Methods to detect Android Malware: static behavioral detection method and dynamic behavioral detection method.

3.2.1. Static Behavioral Detection Method: Through analyzing and comparing the instruction codes of the software, static behavioral detection method detects whether the software contains API function calls which can cause malicious behaviors. Using this method to detect, it firstly acquires Java source codes of Android application software, analyzes whether the software contains sensitive function calls and whether there are security threats, and then comes up with the conclusion whether the software is malicious or not. Static behavioral detection method needs to decompile the application by ways of reverse engineering to acquire source codes. But the analysis is often affected by software encryption and implicit functions (virtual functions, *etc.*,) so it usually cannot draw the correct conclusion.

3.2.2 Dynamic Behavioral Detection Method: Dynamic behavioral detection method works during the running of the application. It detects and records the system's communications, short messages, network interfaces and the network access of the relevant implicit information, thus acquiring the application's behavior model. Dynamic behavioral

detection method can solve the problems that static detection method cannot do because the application codes are encrypted or confused. Dynamic behavioral detection method constructs operation environment by using sandbox, virtual machine and other forms, and simulates the execution of the application to acquire the application's behavior model. It has higher request to the real-time detection.

4. The Behavioral Analysis of Android Malware

When detecting Android malware, whether static detection method or dynamic one, the first step is to acquire the application's way of behaviors including normal applications and malicious software, then it uses machine learning to acquire the characteristic of the malicious software to distinguish the malicious applications from the normal ones.

Focusing on the analysis of the function calls of Android applications, the essay analyzes the function calls of the malicious software to acquire the typical file characteristic of the malicious software, which is used to be the basis of the detection [16].

4.1. Acquiring the Malicious Behaviors

Firstly, the author collects 50 malicious software samples including Trojan horse, spyware and worms, etc., decompiles them, and analyzes the function calls of their APK source codes. In the process of decompilation, the author uses DEX2JAR to transform classes.dex file to Java codes, thus the transformed classes.dex file contains APK implementation codes, acquires its resources file and class file, and uses Java Decompiler to transform the class file to readable format. The binary AndroidManifest.XML file is transformed by AXMLPrinter2 [17].

After all the transformations, the author analyzes the software codes using AndroidManifest.XML file to get all the API calls. The behaviors of 50 malicious software are shown in Table 1.

Table 1. The Statistic of the Samples' Malicious Behaviors

Behaviors	number
Receives SMS/MMS	25
Sends SMS/MMS	25
Send Data over HTTP(s)	23
Uses WiFi	20
Write to disk (internal or external flash card)	20
Obfuscation	20
Send Data (cellular)	19
Receive Data over HTTP(s)	18
Access Device Location	18
Receive Data (cellular)	15
Reads from Disk (internal or external flash card)	14
Can execute commands	12
Mount/Unmount Filesystems	11
Encryption	6
Set Network Properties	6
Send Data (Raw)	5
Receive Data(Raw)	4

After analyzing the behaviors of malicious software in Table 1, we can find that malicious software collects users' private information while having these malicious behaviors, as in Table 2.

Table 2. the Classified Statistic of Private Information Collection

The Name of collect information	number
SMS/MMS	30
IMEI	19
Phone Number	13
Contacts	11
Email	9
Android Version	9
SDK Version	9
Browser History	9
GPS Coordinates	9
Cellular Carrier	7
Data in Flash Card	7
Call Logs	5
Phone Conversations	4
Photos/Videos	3
Root Level	2
Access Point	1

4.2. The Analysis of Malicious Behaviors

After analyzing the behaviors of malicious software, we can conclude the following five forms of malicious software: (1) Malicious fee deduction; (2) Remote control; (3) Stealing information; (4) Rates consumption; (5) Rogue actions.

The relevant permissions of the called systems to the above malicious behaviors are shown in Table 3.

Table 3. The Main Permissions to Each Malicious Behavior

Malicious Behaviors	Main Permissions
malicious fee-deducting	android.permission.RECEIVE(SEND)_SMS/MMS
	android.permission.READ_SMS
	android.permission.CALL_PHONE
	android.permission.CALL_PRIVILEGED
Remote Control	android.permission.RECEIVE(SEND)_SMS/MMS
	android.permission.READ_SMS
	android.permission.INTERNET
	android.permission.ACCESS(CHANGE)_NETWORK_STATE
	android.permission.ACCESS(CHANGE)_WIFI_STATE
Stealing Information	android.permission.READ_CONTACTS
	android.permission.ACCESS_FINE_LOCATION
	android.permission.ACCESS_COARSE_LOCATION
	android.permission.READ(WRITE)_CALL_LOG
	android.permission.READ_PHONE_STATE
	android.permission.INTERNET
	android.permission.ACCESS(CHANGE)_NETWORK_STATE

	android.permission.ACCESS(CHANGE)_WIFI_STATE
fee consumption	android.permission.INTERNET
	android.permission.ACCESS(CHANGE)_NETWORK_STATE
	android.permission.ACCESS(CHANGE)_WIFI_STATE
rogue behavior	android.permission.INTERNET
	android.permission.ACCESS(CHANGE)_NETWORK_STATE
	android.permission.ACCESS(CHANGE)_WIFI_STATE
	android.permission.INSTALL(DELETE)_PACKAGES

5. Detection and Prevention

Through the above analysis of the software samples, we can acquire the main forms of Android malicious software, as shown in Table 3. The result of the analysis can provide corresponding basis to the detection and prevention of Android malicious software.

5.1. Permission Administration

The application permissions are rigidly set in Android system, and each application runs by ways of process isolation. If an application needs to access the data and directories beyond this application, it will apply to the Android system before installing waiting the users for approval. But most users often do not carefully consider whether the applying permission is reasonable when installing the application, so it cannot make sure that the malicious software will not be normally installed.

The bottom frame of Android system uses Linux kernel, and it can establish relevant access policy to each file in smartphone and store it in the storage space of Linux kernel. Therefore, we can set the permissions according to the importance of files [18]. The permissions of files are set in Table 4.

Table 4. The Set of File Permissions

File name	Permissions
/system/app	read
/system/bin	read
/sbin	Read
/data/app	Read
/data/data	Null
/sdcard	Read|write|rename|create
/cache	Read|write|unlink|create

The above method can prevent smartphones from being attacked by malicious software to a large extent, and protect data of smartphones not to be illegally used. But this method usually affects the running of the normal software when it blocks the access of the malicious software, because the method's shortage is that its establishment of access policy is too extensive.

5.2. Behavioral Detection

At present, behavioral detection is a common method to detect Android malicious software. Both static behavioral detection and dynamic behavioral detection analyze the characteristic of the malicious software, then use machine learning to establish relevant regulations to distinguish the malicious applications from the normal ones. Now the range of

Android application software is continually expanding and its application forms are updated constantly, therefore, to keep its accuracy the behavioral detection method should update constantly in order to adapt new environment. The common way is to combine the behavioral detection to black and white lists, which reduces the complexity of the detection and increases the detecting efficiency.

6. Conclusion

Due to its characteristic of openness, Android platform provides convenience to the development and promotion of the application software, which is an important factor for it to occupy smartphone market. On the other hand, it is just due to the characteristic of openness that the spread of the Android malicious software is far greater than other platforms. Along with the coming of 4G and the improvement of the performances of smartphones, the harm of the malicious behaviors is also increasing, which brings greater challenges to the detection and prevention of Android malicious software. If the security problems of Android platform and the approval mechanism are not improved in the process of the future development, the security problems of Android platform would become another Windows.

ACKNOWLEDGMENT

This research was supported by the Anhui provincial college and university Natural Science Foundation, China (No.KJ2013Z302), the 2013 science foundation of Huainan City Science and technology project, China(No.67), and the 2014 science foundation of Huainan Normal University, China(No. 2014xk24zd).

REFERENCES

[1] "CNCERT/CC.CNCERT/CC Annual Report", **(2013)**, http://www.cert.org.cn/publish/main/upload/File/2013 Annual Report.pdf.2014:53-56.
[2] M. Miettinen and P. Halonen, "Host-based intrusion detection for advanced mobile devices", Information Networking and Applications, vol. 20, no. 4, **(2006)**, pp. 72-76.
[3] W. Enck, P. Gilbert, B. Chun, *et al.*, "TaintDroid: An Information-flow Tracking System for Realtime Privacy Monitoring on Smartphones", Proc. of OSDI'10. Vancouver, Canada: [s. n.], **(2010)**.
[4] A. D. Schmidt, "Smartphone malware evolution revisited: Android next target?", In Proceedings of the 4th International Conference on Malicious and Unwanted Software (MALWARE), Montreal, QC, **(2009)**, pp. 1-7.
[5] A. D. Schmidt, "Detection of Smartphone Malware", Universitatsbibliothek, **(2011)**.
[6] A. Bose, X. Hu, K. G. Shin, *et al.*, "Behavioral detection of malware on mobile handsets", Proceedings of the 6th international conference on Mobile systems, applications, and services. ACM, **(2008)**, pp. 225-238.
[7] W. F. Fei, "Study on detection and protection techniques of mobile phone malicious code under the Android platform", Beijing:Beijing Jiaotong University, **(2012)**.
[8] "Google Project Hosting", Androguard [EB/OL]. [2013-12-09]. https://code.google.com/p/androguard/.
[9] W. Enck, M. Ongtang and P. McDaniel, "On lightweight mobile phone application certification", Proceedings of the 16th ACM conference on Computer and communications security. ACM, **(2009)**, pp. 235-245.
[10] Y. Huan, Z. Yuqing, H. Yupu, *et al.*, "Android malware detection method based on permission sequential pattern mining algorithm", Journal on Communications, vol. 34, no. ZI, **(2014)**, pp. 106-115.
[11] H. Peng, C. Gates, B. Sarma, *et al.*, "Using probabilistic generative models for ranking risks of android apps", Proceedings of the 2012 ACM conference on Computer and communications security. ACM, **(2012)**, pp. 241-252.
[12] S. Lee and D. Y. Ju, "A Novel Method to Avoid Malicious Applications on Android", International Journal of Security and Its Applications, vol. 7, no. 5, **(2013)**, pp. 121-130.
[13] L. K. Yan and H. Yin, "DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis", USENIX Security Symposium, **(2012)**, pp. 569-584.

[14] Z. Yang, M. Yang, Y. Zhang, *et al.*, "Appintent: Analyzing sensitive data transmission in android for privacy leakage detection", Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. ACM, **(2013)**, pp. 1043-1054.
[15] Y. J. Ham, D. Moon, H.-W. Lee, J. D. Lim and J. N. Kim, "Android Mobile Application System Call Event Pattern Analysis for Determination of Malicious Attack", International Journal of Security and Its Applications, vol. 8, no. 1, **(2014)**, pp. 231-246.
[16] F. Yuhui and X. Ning, "The Behavioral Analysis of Android Malware", 3rd International Conference on Next Generation Computer and Information Technology (NGCIT 2014), vol. **(2014)**.
[17] K. Sharma, T. Dand, T. Oh, *et al.*, "Malware Analysis for Android Operating", 8th Annual Symposium on Information Assurance (ASIA'13), vol. 31, **(2013)**.
[18] L. Chang-Ping, F. Ming-Yu, W. Guang-Wei, *et al.*, "Light-weight access control oriented toward Android", Application Research of Computers, vol. 27, no. 7, **(2010)**, pp. 2611-2628.

Authors

Yuhui Fan, Received the B.S. degree in Department of Educational Technology from Anhui Normal University, China in 1999, and the M.S. degree in Department of Educational Information Technology from East China Normal University, China in 2008. He is currently an Instructor in the Department of Computer and Information Engineering at the Huainan Normal University. His research interests include network security, computer networking.

Ning Xu, Received the B.E. degree in Department of Computer Science from Anhui Normal University, China in 2002, and the M.E. degree in Department of Computer Science from Anhui University, China in 2009. She is currently an Instructor in the Department of Computer and Information Engineering at the Huainan Normal University. Her research interests include network security, computer networking.