An Android Malware Detection Method Based on Feature Codes

Yiran Li[1, a], Zhengping Jin[2, b]

[1]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China;

[2]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China.

[a]yiranlee1991@163.com, [b]zhpjin@bupt.edu.cn

Keywords: android security, malware, feature codes, system call, machine learning.

Abstract. The Linux-based android operation system is now exposed to high risks of security since the malware of smart phone explodes. For the purpose of effectively detecting the malware on the android platform, an android malware detection method based on feature codes is described in this paper. By using the function call and system call, analyzed and extracted from the malware sample library, as the feature vectors which will be subject to training and classification upon machine learning and data mining algorithm, a feature library and a detection model is established. An android malware detection system, ANDect, is developed upon this method and used for detecting 350 malicious applications and 750 non-malicious applications. As the results, ANDect is proven that it can effectively find out the undiscovered malicious Applications of android by utilizing the feature vectors of codes from the android applications, with high accuracy and low false positive rate.

Introduction

With the popularity of android smart phones, the detection and analysis of related malware come into focus in recent years [1].

At present, the malicious code detecting methods for android apps can be classified into the signature-based and behavior-based detections. Most of security software providers adopt the antivirus measures mainly by using the malware detecting methods based on signature[2].The open source tool, Androguard[3], also adopts the signature-based methods.The main drawback of this method is dependent on signature library ,and always accompanied with the failures to detect the unknown malicious apps. Crowdroid focuses on the critical system call of kernel and analyzes the malicious behaviors by collecting the behavior features of system calls from the device[4]. Droidscope establishes the static behavior senses for both OS and Java layers synchronously based on system call and Java call, and realizes better detection of malicious software[5]. These two use behavior-based detection to classify.

In recent years, many researches had been made on detecting the malicious codes of android by using machine learning and data mining technologies. In 2011, Iker and others adopted the dynamic method to extract the system call features, and used the K-means clustering algorithm to distinguish the malicious codes within the kin programs[5], but this method is only applicable for detecting the malicious codes in the repacked program. In 2012, Shabtai and others extracted API features manually by using the dynamic method, and adopted the sorting algorithm to detect the malicious codes programmed by themselves for android[6], instead of using the existing apps for verifying.

To achieve more effective detection of malicious codes on android platform, this paper provides an android malware detection method based on feature codes. Firstly, the features of sensitive function calls will be extracted from a sample database of known android malware and used as the features of the codes in the application layer; secondly, the system calls of apps including native codes will be obtained by using both dynamic and static methods and used as the features of the codes in the native layer; finally, these two kinds of features of codes will be used as the feature vectors which are subject to the model training based on machine learning in order to obtain the

detection model. An android malware detection system, ANDdect, is presented in this paper, including the assessment and validation of the detecting results of this system.

Feature Extracting Algorithm

Both dynamic and static behavior analyzing techniques are adopted here to extract the two different features of android apps, as described as follows.

Malicious Function Call Features.

Android provides APIs for application developers. A developer can realize various application behaviors by calling APIs, and the malicious behaviors are the application behaviors consisting of a series of sensitive APIs at high risk levels. Therefore, when such sensitive APIs to be called by malicious behaviors more frequently are selected from Android SDK and numbered, such malicious behaviors can be expressed as the feature vectors by using API arrays.

Based on the above, a concept of feature vector space of malicious behaviors is provided here to put the malicious behaviors in a feature vector space which is applicable to and includes all kinds of malware.

Use

$$F_{malware} = \{f_1, f_2, \ldots, f_k\} \quad (1)$$

to represent the feature set of the samples in the malicious sample database, wherein, f_i is the Behavior feature vector of a malicious app, $1 \le i \le k$.

For any malware, define a feature vector consisting m sensitive API calls, provided that

$$f_i = [a_1, a_2, \cdots, a_m] \quad (2)$$

Wherein, a_j is limited by $a_i = p$, to indicate the times of call of this sensitive API function a_j by this malware f_i.

Features of Unsafe System Call by Malicious Behaviors.

Linux is the lower layer of android, and each request from the upper layer needs the system calls to realize the operation on the Linux Kernel[7]. Since the system calls are the interfaces between user's controls and the kernel, android apps can complete the network access, file operation, SMS message sending and other operations through the system calls. As for android apps using native codes, the .so file under the lib folder is related to android native. It can, together with other shared object files or re-locatable files, generate ELF object files or process images, provided that such function calls shall be realized through external calls, shared lib or system calls[8].

Therefore, the features of malicious behaviors of an android app, especially an android app including native codes, can be characterized by the times and sequences of linux system calls. In addition, any confusion or encryption cannot change the sequence of system calls by android apps, so that the attacks utilizing confusion or encryption can be ruled out effectively in this way.

Accordingly, as for the malicious samples in the database, use the vector

$$T_{native} = \{t_1, t_2, \cdots, t_k\}, \quad (3)$$

to represent the sample feature set of system calls by the malicious samples, wherein, t_i is the feature vector of malware *i*, with $1 \le i \le k$.

For any malware, define the geature vector

$$t_i = [B_i, S_i^{'}], \quad (4)$$

to indicate the features of unsafe system calls by this malware, where, B_i is a feature vector including n dangerous system function calls, and expressed as

$$B_i = \{b_1, b_2, \cdots, b_n\}, \quad (5)$$

with b_j limited by $b_j = q$, to indicate the times of system function call by malware t_i, and $S_i^{'}$ is the sequence of dangerous system function calls malware t_i.

Android Malware Detection System

Based on the above extracting algorithms of malicious behavior features, an android malware detection system, ANDect, is proposed here as shown in Fig 1.

This system five components: app sample database, preprocess modular, feature extracting modular, feature set and model training modular, wherein, app sample database mainly includes the malware samples provided by open source community Contagio, and safe apps, passed virus examinations, obtaied by crawlers from Google Play and other third party app stores; preprocess modular unzips and recompiles the samples to obtain the smali intermediate codes and .so files necessary for the extraction of feature vectors; feature extracting modular can obtain the function call features and unsafe system call features through static analysis on the feature codes of samples based on the above feature extracting algorithms, in addition to the system call sequence and frequencies by Strace[9] in a dynamic way; feature set is the sample feature vector set obtained by combining these two kinds of feature vectors, which are to be put into the model training modular to complete the model training through the repeated learning based on SVM algorithm.

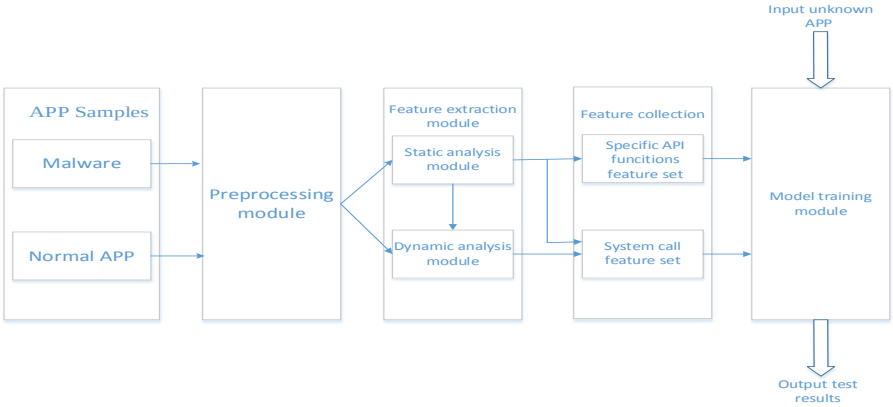

Fig.1 ANDect Test System Architecture Diagram

Experiment and Results

Design.

Samples used include 350 malicious samples and 750 normal samples, and 90% of these two kinds of samples are selected randomly as training set, with the rest as test set.

In this experiment, APIs, related to the typical sensitive function calls marked among thousands of android API functions, are 130 in 6 categories. For example, No. 2EED7318CA564A909E75AD6A6CAD5CDE[10] malicious sample is scanned, and 14 typical marked APIs are found in smali codes, with call information obtained as shown in Fig 2, including times of calls.

```
[2EED7318CA564A909E75AD616CAD5CDF] :
  0 : Landroid/telephony/TelephonyManager;->getLine1Number()Ljava/lang/String;
  7 : Ljava/net/HttpURLConnection;->connect()V
  0 : Landroid/net/wifi/WifiManager;->getWifiState()I
  1 : Landroid/location/LocationManager;->requestLocationUpdates(Ljava/lang/String;JFLandroid/location/LocationListener;Landroid/os/Looper;)V
  0 : Landroid/telephony/TelephonyManager;->getSubscriberId()Ljava/lang/String;
  3 : Landroid/location/Location;->getProvider()Ljava/lang/String;
  0 : Landroid/net/wifi/WifiManager;->getScanResults()Ljava/util/List;
  2 : Landroid/location/LocationManager;->getBestProvider(Landroid/location/Criteria;Z)Ljava/lang/String;
  0 : Landroid/telephony/TelephonyManager;->getCellLocation()Landroid/telephony/CellLocation;
  0 : Landroid/telephony/TelephonyManager;->getSimSerialNumber()Ljava/lang/String;
  2 : Landroid/telephony/TelephonyManager;->getDeviceId()Ljava/lang/String;
  0 : Landroid/telephony/SmsManager;->sendTextMessage(Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;Landroid/app/PendingIntent;Landroid/a
  0 : Landroid/net/wifi/WifiManager;->startScan()Z
  3 : Landroid/location/LocationManager;->getLastKnownLocation(Ljava/lang/String;)Landroid/location/Location;
```

Fig. 1 Part of the API function call features scan results of some sample

Linux kernel has about 290 system calls in total, but not all of them can be used for describing the malicious behaviors of an app. To reduce the calculating and processing workload, 45 system functions in 5 categories are selected in this paper since they are the most representative functions for describing the potential malicious behaviors[11].

Choose the same malware which mentioned above as the example, then extract dangerous system calls and partial results are showed in Fig 3:

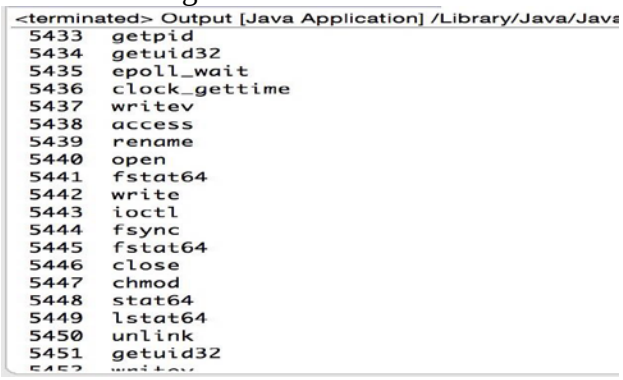

Fig.3 Part of the Dangerous system call features scan results of some sample

Assessment.

Different amounts of normal and malicious samples are selected to validate the effectiveness of ANDect, as listed in Table 1. When the Sample Set is 100, the false negative number is 35, with False Negative Rate(FNR) at 0.350,False Positive Rate(FPR) at 0.120 and Accuracy at 0.765. Accuracy can up to 0.880 when the sample set is 350. As shown in the Table, the FNR and FPR are decreasing along with the increasing of sample set, but the accuracy rate is increasing. In addition, the FPR is lower than the FNR since ANDect conducts the detection on android apps based on feature codes.

Table 1 Test results of the ANDect system

Nomal	malware	FN	FP	FNR	FPR	Accuracy
100	100	35	12	0.350	0.120	0.765
200	200	45	19	0.225	0.095	0.840
300	300	53	22	0.176	0.073	0.875
350	350	59	25	0.169	0.071	0.880

The well-known open source tool, Androguard, is used for comparing with ANDect upon 1100 same Samples, with the results of comparison listed in Table 2. Androguard can detect 298 malicious samples at detecting rate at 0.851, while ANDect can find 319 malicious samples at detecting rate of 0.911, which means ANDect is obviously better than Androguard.

Table 2 Compared ANDect with Androguard

	Normal	Malware	Accuracy	APKs
Androguard	750	350	0.899	1100
ANDect	750	350	0.911	1100

Conclusion

This paper proposes a feature extracting algorithm, and according to this implemented a system for testing. The sensitive function calls and unsafe system calls are analyzed based on the feature codes by using the known sample database, and are used as the feature vectors trained by machine learning algorithm to build the android malware detection system, ANDect. It is proved that the ANDect based on feature codes can detect the unknown malware effectively by using 350 malicious apps and 730 normal apps. In the future, simplify redundant vectors and incorporate more types of behavior features is the next research direction, so that we can put the efficiency and speed of detection into a new level.

Acknowledgements

This work is supported by NSFC (Grant Nos. 61300181, 61502044), the Fundamental Research Funds for the Central Universities (Grant No. 2015RC23).

References

[1] Cao Y, Liu J, Miao Q, et al. Improved behavior-based malware detection algorithm with AdaBoost[J]. Journal of Xidian University, 2013, 40(6):116-124.

[2] Natani P, Vidyarthi D. An Overview of Detection Techniques for Metamorphic Malware[M]//Intelligent Computing, Networking, and Informatics. Springer India, 2014: 637-643.

[3] Information on: http://code.google.com/p/androguard/

[4] Burguera I, Zurutuza U, Nadjm-Tehrani S. Crowdroid: Behavior-Based Malware Detection System for Android[C]// Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices. ACM, 2011:15-26.

[5] Yan L K, Yin H. DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis[J]. Proceedings of Usenix Security Symposium, 2012:29-29.

[6] Shabtai A, Kanonov U, Elovici Y, et al. “Andromaly”: a behavioral malware detection framework for android devices[J]. Journal of Intelligent Information Systems, 2012, 38(1):161-190.

[7] Zheng M, Sun M, Lui J. DroidTrace: A ptrace based Android dynamic analysis system with forward execution capability. Wireless Communications and Mobile Computing Conference (IWCMC), 2014 International. IEEE, 2014: 128-133.

[8] Yang H, Zhang Y Q, Yu-Pu H U, et al. A Malware Behavior Detection System of Android Applications Based on Multi-Class Features[J]. Chinese Journal of Computers, 2014.

[9] Information on: http://sourceforge.net/projects/strace/

[10] Information on: http://contagiominidump.blogspot.com/

[11] Wang Z Q, Zhang Y Q, Liu Q X, Huang T P. Algorithm to detect Android malicious behaviors. Journal of Xidian University,2015,42(3)