ADetect: Hybrid Analysis Feature Extraction for Android malware Detection

Kyaw, May Thu
University of Computer Studies Yangon

Soe, Yan Naung
University of Computer Studies Yangon

Kham, Nang Saing Moon
University of Computer Studies Yangon

https://doi.org/10.15017/1960660

出版情報：Proceedings of International Exchange and Innovation Conference on Engineering & Sciences (IEICES). 4, pp.27-31, 2018-10-18. 九州大学大学院総合理工学府
バージョン：
権利関係：

ADetect: Hybrid Analysis Feature Extraction for Android malware Detection

*May Thu Kyaw[1], Yan Naung Soe[1], Nang Saing Moon Kham[1]
[1]University of Computer Studies Yangon.
maythukyaw@ucsy.edu.mm

Abstract: *Today, mobile devices are very popular in everyday life of our sociality for communication or many perspectives. They become an interesting point for malicious attackers. Malicious software which can destroy mobile devices or steal sensitive information are growing in every form of people's live. A number of researches have been proposed to detect malicious software in recent year. However, they still suffer with many intelligent malicious software that a traditional methodology is not sufficient to detect the key features of intelligent malware. To address this problem, this paper proposes a feature extraction method from Android malware applications using hybrid analysis method to improve Machine Learning based detection framework. ADetect can achieve 80% detection accuracy, which are tested by using Random Forest, K-nearest Neighbor and Naïve Bayes (NB) classifiers.*

Keywords: Mobile, Malware, Machine Learning

1. INTRODUCTION

A recent report of Gartner[1], an American information technology research and advisory firm, android has become the No.1 operating system in 2013 and also dramatically transcend a large number shipments of its devices in 2014. In contrast to other platforms, Android grants the installation of applications(.apk) from various sources, such as Google Play Store and other mediator markets. As a result, it has led to an increase in their potential as a target for malicious activities.

In the internet age, privacy and security issues are developed rapidly for mobile computing because of mobile devices take place in computing platforms and data storage units. Malware developers used various intelligent methods to overcome traditional and modern malware protection and detection mechanisms. Therefore, analysis of malware and detection approaches have become an active area of research. Many researchers proposed a number of techniques such as graph theory [2], machine learning [3,4] and information visualization [5,6] to hinder the growing amount and sophistication of Android malware.

Malware analysis can be categorized in three methods: static, dynamic, and hybrid analysis techniques. Static analysis is based on extracting features of application without running. It checks out an application's manifest (AndroidManifest.xml) and disassembled code. Reversely, dynamic analysis methods analyze the application's behavior during the execution process. Hybrid method is the combination of static and dynamic analysis. However, there has a challenge for accuracy in malware characterization and detection, mainly in day-by-day changing of intelligent malware and the open distribution channels of Android apps

To overcome this situation, we proposed a framework with hybrid analysis, namely ADetect (**A**ndroid **Detect**or), that can automatically detect whether an apk is a malware or not. ADetect use marketplace crawlers, filtering and feature extraction and classifier. It means that we use all apk(malware or benign) to process, and then filter out which are either known malware or not. This paper emphasizes on the feature extraction for malware detection. We propose a hybrid security solution, integrated static and dynamic analysis method, to analyses and characterize an unknown executable file. The rest of the paper is structured as follows. Section 2 presents the motivation of this paper. Section 3 provides the literature review. The proposed system illustrates in Section 4. Finally, Section 5 concludes and discuss future work to detect of android malware.

2. MOTIVATION

Today, mobile devices have become a widely used for personal and business purposes. The ecosystem of Android application has increased dramatically in recent year. Over 3 million apps currently available at Google Play official market [14]. Mobile device became a pool of data for us and it may carry sensitive data, such as credit card account number, username, password, etc[7]. Smartphones may now represent an ideal target for malware writers and it has become the most coveted and viable target of malicious apps.

Our present study aims at designing and developing better approach to detect malicious application in Android devices. More precisely, ADetect, a framework for detection of Android malware based on Machine Learning technique. There are various elements such as network, permission, method call, java code and behavior of application etc. The selection of useful feature from large number of available can change the result of the whole experiment (Guyon and Elisseeff, 2003). The following are the benefits of feature selection:

- Reducing the size of dataset can easily visualize the trend in data (Crussell et al.).
- There is huge amount of data in analyzing datasets. Therefore, compressing them to only useful feature save not only time but also save money. It also reduces for the time of real world implementation (Crussell et al.).
- Feature selection help to get accurate results of machine learning algorithms because it removes noisy and irrelevant data from datasets (Jensen and Shen, 2008).
- Feature selection also enhance model simplification that can make easier to interpret by researchers or users.

3. LITERATURE REVIEW

The process of Machine learning algorithms is learning the patterns from the data. Feature extraction is the first step of every machine learning algorithm in malware analysis. There are many approaches for mobile malware detection and analysis, such as static analysis, dynamic analysis and hybrid analysis approach for malware detection.

DroidRanger [8] is a one of hybrid analysis using manifest file and bytecode and monitor during execution. It is a footprint-based detection engine that extracts features such as permission and semantic word in bytecode (e.g. INTERNET) for static analysis and also on a heuristics-based detection engine that monitors applications during their execution for dynamic analysis, e.g., system calls with root privileges.

Shina Sheen, R.Anitha, V.Natarajan [9] uses different features vector for example API call feature and permission based features to consider for a better detection. They use collaborative approach based on probability theory. Kabakus Abdullah Talha, Dogru Ibrahim Alper, Cetin Aydin proposed a method based on permissions used in an application and static analysis is made using machine learning algorithm such as logistic regression [10].

Seung-Hyun Seo, Aditi Gupta, Asmaa Mohamed Sallam, ElisaBertino, KangbinYim proposed a method to detect mobile malware threats to homeland security. In their proposed approach, they define different characteristics of android malware and provide a case study which are feasible against Homeland Security. They used DroidAnalyzer which is static analysis tool for identification of vulnerabilities in android applicaions and the presence of root exploits [11].A. Shabtai, L. Tenenboim-Chekina, D. Mimran, L. Rokach, B. Shapira, Y. Elovici, discovered a method to find mobile malware based on semi supervised machine learning despite of regular static and dynamic base analysis[12].

Wanqing You, Kai Qian, MinzheGuo, Prabir Bhattacharya proposed a hybrid approach for mobile threat analysis. The key of this approach is the unification of data states and software execution on critical test paths conditions. The outcome leads to combine the benefit of static and dynamic analysis. This is the main benefit of their technique that is they used a hybrid approach for analysis [13].

4. SYSTEM ARCHITECTURE

In this section, we first introduce the overall architecture of ADetect and then describe each module individually to explain how ADetect works for Android malware detection. Figure 1 illustrates the experiment work flow structure consisting of four phases.

The first stage is data collection, which collects normal and malicious applications. In the second phase is feature extraction and selection. In this stage, extracted features are selected, labelled and stored to be applied in the next phase. The Machine Learning classifiers entail the third phase, whereby the stored information trains the Machine Learning classifiers to produce several detection models. The last phase is the evaluation and choice of a classifier based on empirical data obtained, in order to build our framework.

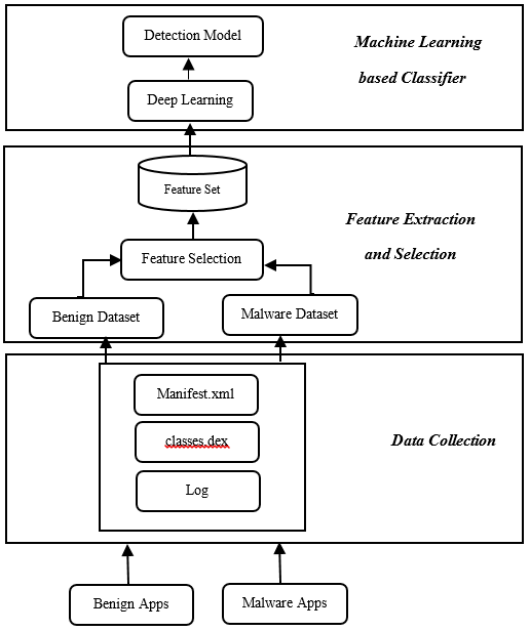

Fig. 1. System Architecture

4.1 Data Collection

In this data collection phase, we use two main approaches to collect the data. Firstly, we crawl malware samples directly from well-known Android malware blogs such as Contagio Mobile Malware Mini Dump [15]. Because of no standard dataset for benign application, we collected dataset from Google Play Store [14] which is considered as the official market with the least possibility of malware application. We have collected total 219 applications from various sources. Table 1 gives the malware families chosen for this experiment.

Table 1. Malware Dataset Description

No	Malware Family Name	Total	Characteristics
1	DroidDream	21	Hijacks application and controls the UI and performs commands received from a hacker
2	DroidKungFu3	30	Malicious code is encrypted and it steals user's phone number and send it to hacker
3	DroidKungFu4	20	C&C server address is in the native program but in cipher text. It receives commands from a hacker
4	Geinimi	39	Makes phone calls in background. Commands are received from a hacker
5	Anserver Bo	13	Silently downloads an update for malicious application on run time containing malicious code from a hacker
	Total	113	

4.2 Feature Set

We proposed 7 features for static and dynamic analysis which are extracted from our dataset (malicious or benign). These features are used permissions, requested permissions, permission request APIs, network APIs, suspicious calls, providers and instruction sequences.

Hybrid Analysis Features

Used permissions: Most of the Android applications request many permissions without using it. They use only a subset of the requested permissions. We can get more exact observation of apps intension by extracting the used permissions. Eg.,`<uses-permission android: name="android.permission.ACCESS_WIFI_STATE"/>`

Requested permissions: Most of user are easily grant the permissions without knowing of how it is work. Permission plays an important role in security of Android operation system. Most of user are easily grant the permissions without knowing of how it is work. An application can install itself by user granting and can perform malicious behaviors. Eg., `INTERNET, WRITE_EXTERNAL_STORAGE AD CONTACT.`

Permission request APIs: API calls can be requested Android permission. For example, a sendDataMessage call requests permission SEND SMS and to receive SMS, developer use `android.permission.RECEIVE_SMS.`

Network APIs: Malwares are now try to access the network and then send out sensitive data by using network APIs.

Suspicious calls: These Suspicious API calls such as communicating over the network, sending and receiving messages, and executing external commands are frequently used by malware developers.

Providers: The provider can be used to manage structured access to data storage. `<provider>` element is used to define provider in the manifest file. Otherwise, the system is not assuming as a provider and doesn't run them.

Instruction sequences: Open source tool is used for extraction of low level instructions (also known as Dalvik bytecode) from an application.

Table 2. Overview of Features

Type	Features (Keywords)
API calls related	getSubscriberId;getLine1Number; getSimSerialnumber; SMSReceiver; getNetworkOperactor;Contacts; FindClass; KeySpec; getCellLocation; onActivityResult;
Permission	ACCESS_FINE_LOCATION; WRITE_SMS; WRITE_ CALL_LOG; SEND_SMS;WRITE_APN_SETTINGS; RECEIVE_BOOT_COMPLETED; RECEIVE_MMS;BROADCAST_SMS ;INTERNET;ACCESS_COARSE_LOCATION;

Feature Vector

We used machine learning algorithms for characterization of android malware because most of them process data with numerical vectors. Therefore, we first need to map our extracted features into a joint feature vector.

Suppose V be a vector for all of selected features vector. Let i^{th} be every application in Android apps dataset. So, we generate $V_i = \{v_1, v_2, \ldots, v_n\}$ and

$$V_n \begin{cases} 1, & \text{if } n^{th} \text{ permission exits} \\ 0, & \text{otherwise} \end{cases}$$

In our system permission features are stored in binary format (0 and 1) in separation of comma where 1 define the permission is exist and 0 for not exist of that permission. In addition, we define a variable D, where D ∈ {benign, malware}. We represent the variable to 1 for benign and -1 for malware application. Figure 2 and 3 are the example of vector for these benign and malware apk:

0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,
0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,**1**

Fig. 2. Permission Vector of Benign App

0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,
1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,
1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,**-1**

Fig. 3. Permission Vector of Malware App

4.3 Feature Extraction and Selection

All of android applications are developed in zip format that we know apk format. Features are encrypted in that apk file for example permissions, Network API, IP address and URL. We combine static and dynamic analysis method to extract that features which are assisted in characterization of benign or malware as shown in Fig 2 and 3. There are three types of feature in our proposed system: required permission, dynamic behaviors and sensitive APIs. Among them, dynamic behaviors are extracted with dynamic analysis, whereas sensitive APIs and required permissions are extracted with static analysis.

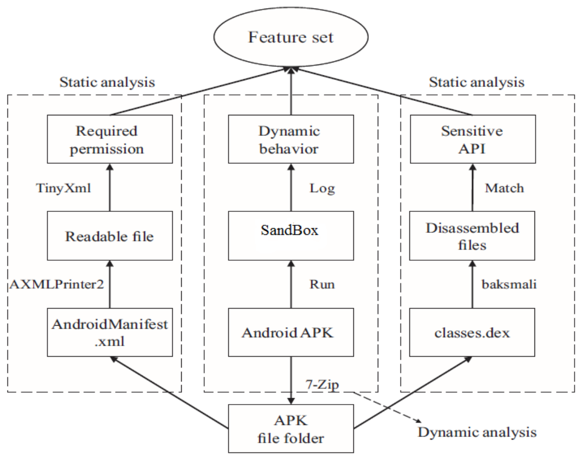

Fig.2. Proposed Feature Extraction for apk

In the static phase, we decompress the apk file with the 7-Zip tool to retrieve the content. As a result, AndroidManifest.xml and classes.dex are obtained. Our system used AXML-Printer2 for retrieving the required features from AndroidManifest.xml. We can also obtain the permissions required by the app with the TinyXml parser. For eg., android.permission.READ_SMS need permission for reading SMS and android.permission. network is the permission that required to access the network from app. We got 120 permissions in this step.

Moreover, we also obtained classes.dex file in decryption of apk. Our system uses disassembler basksmali for parsing that file. In that way what API

function are worked. For example, chmod, is a sensitive API because that can change the role of user permission on files. We earned total 59 sensitive API function in this step.

We use DriodBox[16] for installing and running android apk in dynamic analysis phase. DroidBox can execute at the application framework level because it is one kind of SandBox. Therefore, it can analyze a dynamic taint analysis with system hooking and then it can also monitor a number of app behaviors such as Short Message Services (SMS), information leaks, cryptography operations, network and file input/output, and mobile phone calls. In this study, we can monitor a total of 13 app dynamic behavior. For instance, `action_sendnet` is the action that sends data over the network and `android.permission. ACCESS_FINE_LOCATION` is the action that sends victim's location to the server.

As a conclusion, we totally earned 192 features of each application by using static and dynamic analysis. In this case, each feature represents as binary number, where denote 1 for a feature occur: its feature value is 1; otherwise, its feature value is 0.

4.4 Machine Learning based Classifier

Traditional way of machine learning models such as Support Vector Machine are processed with less than three layers of computation units. Therefore, they can be considered as shallow architectures. Fortunately, deep learning models is not same as that situation because of a deep architecture. A deep learning model are developed with different deep architecture [18] in applied environment such as convolutional neural network and Deep Belief Network(DBN). In our system, we propose DBN architecture to develop our deep learning model and characterize Android apps. For our proposed framework, we have two process phases for a deep learning model construction, pretraining and back-propagation. The pre-training phase are constructed in unsupervised manner. Because of Restricted Boltzmann Machines (RBM) our DBN is hierarchically built with the deep neural network take a notice of a hidden variable model. The way of process is effective for high-level representations of our features. In the supervised back-propagation phase, we finetuned the sample with labeled with the pre-trained DBN in a supervised manner. The same dataset (application set) are used in both of training process of deep learning model. The deep learning model will be completely built in this way.

5. EXPERIMENT AND EVALUATION

This section provides the experiments performed and the results obtained from our proposed system. Three cases are evaluated in this section. Firstly, static analysis was carried out using only the permission data for training and testing. Secondly, dynamic analysis (system call frequency data) was analyzed for training and testing. Lastly, training and testing was carried out by combining static and dynamic such as the permission data and system calls frequency data.

Accuracy of a test is evaluated on how well the test is able to distinguish between a malware and benign. We use three classifiers Random Forest (RF), K-nearest Neighbor (KNN) and Naïve Bayes (NB) to evaluate our hybrid feature selection method. The evaluation was performed by measuring the following equation:

$$\textbf{True positive rate} = \frac{TP}{(TP + FN)} \quad (1)$$

$$\textbf{False positive rate} = \frac{TP}{(TN + FP)} \quad (2)$$

$$\textbf{Precision} = \frac{TP}{(TP + FP)} \quad (3)$$

$$\textbf{Recall} = \frac{TP}{(TP + FN)} \quad (4)$$

$$\textbf{F-measure} = \frac{2 \times \mathrm{Re}\,call \times \mathrm{Pr}\,ecision}{(\mathrm{Re}\,call + \mathrm{Pr}\,ecision)} \quad (5)$$

$$\textbf{Accuracy} = \frac{TP + TN}{(TP + TN + FP + FN)} \quad (6)$$

We use Weka[17], which is a collection visualization tools. It also have many algorithms for data analysis and predictive modeling. Moreover, it has a graphical user interface which can help easy access to analysis and algorithm.

Table 4. Experimental Results with ML-based classifier

Feature Set	Classifier	True Positive Rate	True Positive Rate	Accuracy (%)
Static	RF	0.872	0.08	89
	KNN	0.795	0.358	68
	NB	0.812	0.6	60
Dynamic	RF	0.634	0.491	57
	KNN	0.094	0.244	83
	NB	0.828	0.674	58
Hybrid	RF	0.896	0.108	89
	KNN	0.824	0.232	79
	NB	0.813	0.492	72

As a result, the system calls frequency results (dynamic analysis) were not as effective as the permissions data. However, the effect of combining both the feature vector fetched a better result as shown in Table 4. All experiments are performed on a 3.40GHz Intel Core i7 PC with 4GB physical memory, using Weka and MS Windows 10.

6. CONCLUSION AND FUTURE WORK

In this paper, we provide a detecting architecture aiming at identifying harmful Android applications without modifying the Android firmware. We proposed hybrid feature selection method by addressing the selecting of key features from Android apps. Three Machine Learning classifiers is used to evaluate malware classification accuracy in our feature set. According to the result, our approach can be considered as an effective approach in malware detection. However, the time is too long in real smart phone because of hardware requirement. A major benefit of the approach is that the system is designed as platform-independent so that smart devices with different versions of Android OS can use it. Many commercial antivirus software use signature-based approach which is static analysis method and cannot effectively handle malware with code obfuscation technique. For future work, we design our system to develop a real-time malware detection infrastructure.

7. REFERENCES

[1] Gartner website: http://www.gartner.com/newsroom. (assessed 12.04.18)

[2] Elhadi AAE, Maarof MA, Barry BI, Hamza H. Enhancing the detection of 16 metamorphic malware using call graphs. Comput & Sec 2014; 46: 62–78. 17

[3] Rieck K, Trinius P, Willems C, Holz T. Automatic analysis of malware behavior 20 using machine learning. J Comput Sec 2011; 19: 639–668. 21

[4] Nataraj L, Karthikeyan S, Jacob G, Manjunath B. Malware images: visualization 1 and automatic classification. Proc of VizSec'11 2011; 4. 2

[5] Nataraj L, Karthikeyan S, Jacob G, Manjunath B. Malware images: visualization 1 and automatic classification. Proc of VizSec'11 2011; 4. 2

[6] Saxe J, Mentis D, Greamo C. Visualization of shared system call sequence 3 relationships in large malware corpora. Proc of VizSec'12 2012; 33–40.

[7] Y. Wang, K. Streff, and S. Raman, "Smartphone Security Challenges," Computer (Long. Beach. Calif)., vol. 45, no. 12, pp. 52–58, Dec. 2012.

[8] T.Zhou, Z.Wang, W.Zhou, and X.Jiang. "Hey you, get off of my market: Detecting malicious apps in official and alternative android markets", in Proc. Of Network and Distributed System Security Symposium (NDSS 2012), San Diego; CA, USA, Feb 2012.

[9] Shina Sheen, R.Anitha,V.Natarajan, "Android based malware detection using a multifeature collaborative decision fusion approach", October 2014, Elsevier, Neurocomputing151(2015)905–912.

[10] Kabakus Abdullah Talha , Dogru Ibrahim Alper , Cetin Aydin , "APK Auditor: Permission-based Android malware detection system", March 2015, Elsevier, Digital Investigation 13 (2015) 1-14.

[11] Seung-Hyun Seo, Aditi Gupta, Asmaa Mohamed Sallam, ElisaBertino, KangbinYim, "Detecting mobile malware threats to home land security through static analysis", June 2013,Elsevier, Journal of Network and Computer Applications 38(2014)43–53

[12] A. Shabtai, L. Tenenboim-Chekina, D. Mimran, L. Rokach, B. Shapira, Y. Elovici, "Mobile malware detection through analysis of deviations in application network behavior", Feb 2014,Elsevier, computers & security 43(2014)1-18

[13]Wanqing You, Kai Qian, Minzhe Guo, Prabir Bhattacharya, "POSTER: A Hybrid Approach for Mobile Security Threat Analysis",June 2015,ACM, ACM 978-1-4503-3623-9/15/06.

[14]Google PlayStore. Available: https://play.google. com/store

[15] Malware Repository website: http://contagio minidump.blogspot.com. (accessed 21.07.17)

[16] DroidBox: An Android application sandbox for dynamic analysis, http://www.honeynet.org/ gsoc2011/slot5, 2015.

[17]Weka the University of Waikato website: https://www.cs.waikato.ac.nz/ml/weka/ (assessed 02.08.18)

[18] Y. Bengio, Learning deep architectures for ai, Foundations and Trends in Machine Learning, vol. 2, no. 1, pp. 1–127, 2009.