Machine Learning Based Identification for Android Malicious Applications

Zarni Aung[1] and Win Zaw[2]

[1]Faculty of Information and Communication Technology, University of Technology (Yatanarpon Cyber City), Pyin Oo Lwin, Myanmar

[2]Department of Information Technology, Technological University (Thanlynn), Thanlynn, Myanmar

ABSTRACT

Mobile malware is rapidly becoming a serious threat. There are many different types of mobile malwares in real world and they depend on the platforms or operating systems which are installed in mobile devices. The Fake Player Trojan is the first malware for android platform that was discovered in 2010. The number of android malicious applications has been consistently rising because android phones are widespread and steadily gaining popularity. Therefore, android malicious applications detection has become a popular research area. We propose a framework to identify different types of android malicious applications by using machine learning approaches in this paper. This framework intends to extract features from android applications and will identify malware application type with better accuracy results. In an evaluation with 1,000 applications, the proposed framework identifies malicious applications with accuracy rate over 90% and few false alarms.

KEYWORDS

Malware, Android, Machine Learning, Malicious Applications

1. INTRODUCTION

Malware - short for malicious software - is everywhere. In various forms for a variety of incentives, malware exists on desktop PCs, server systems and even mobile devices like smart phones and tablets. Some malware litter devices with unwanted advertisements, creating ad revenue for the malware creator. Others can dial and text so-called premium services resulting in extra phone bill charges. Some other malware is even more insidious, hiding itself (via rootkits or background processes) and collecting private data like GPS location or confidential documents.

The growing popularity of smartphones and tablet computers has made mobile platforms a prime target for attack. In a recent study conducted by Juniper Networks, the number of malicious mobile applications observed in the wild is reported to have grown exponentially at a rate of 614% between March 2012 and March 2013 [14]. To mitigate these security threats, various mobile-specific Intrusion Detection Systems (IDSes) have been recently proposed. Most of these IDSes are behavior-based, i.e. they don't rely on a database of malicious code patterns, as in the case of signature-based IDSes. In this paper, we describe a machine learning based malware identification of android applications for android phones users. We then apply standard machine learning classification algorithms to identify android malware applications. Our results indicate that relatively simple classification algorithms can identify malware with over 90% accuracy and few false positives.

The reminder of the paper is organized as follows. Section 2 lists some related work. Section 3 discusses the different types of malware and Section 4 briefly describe about machine learning. Section 5 presents the proposed framework and Section 6 discusses experiments of malware identification architecture and concludes the system and proposes future work in Section 7.

2. Related Work

It is a straightforward idea to detect a harmful mobile application based on the permissions it requests. [13] attempts to explore the possibility of detecting malicious applications in Android operating system based on permissions. In addition to the requested and the required permissions, this study extracts several easy-to-retrieve features from application packages to help the detection of malicious applications. Four commonly used machine learning algorithms including *AdaBoost*, *Naive Bayes*, *Decision Tree (C4.5)*, and *Support Vector Machine* are used to evaluate the performance. Machine learning classification has been widely used in malware detection [1-5]. Several approaches [6, 7] have been presented that focus on classifying executables specifying the malware category; e.g., Trojan horses, worms, viruses, or even the malware family. Regarding Android, the number of new malware samples is also increasing exponentially and several approaches have already been proposed to detect malware. Shabtai et al. [8] built several machine learning models using as features: the count of elements, attributes and namespaces of the parsed Android Package File (.apk). To validate their models, they selected features using three selection methods: Information Gain, Fisher Score and Chi-Square. Their approach achieved 89% of accuracy classifying applications into only 2 categories: tools or games.

There are other proposals that use dynamic analysis for the detection of malicious applications. Crowdroid [9] is an approach that analyses the behavior of the applications. Blasing et al. [10] created AASandbox, which is a hybrid dynamic-static approximation. The dynamic part is based on the analysis of the logs for the low-level interactions obtained during execution. Shabtai and Elovici [11] also proposed a Host-Based Intrusion Detection System (HIDS) which uses machine learning methods that determines whether the application is malware or not. Google has also deployed a framework for the supervision of applications called Bouncer. Oberheide and Miller [12] revealed how the system works: it is based in QEMU and it performs both static and dynamic analysis. In light of this background, we present MADS (Malicious Android applications Detection through String analysis), a novel approach for detection of malware in Android. This method employs the strings contained in the disassembled Android applications, constructing a bag of words model in order to train machine-learning algorithms to provide detection of malicious applications.

3. Malware

Malware is software created by an attacker to compromise security of a system or privacy of a victim. A list of different types of malware is listed in Table 1.

Table 1: Categories of Malware

Malware	**Brief Description**
Worm	Malware that propagates itself from one infected host to other hosts via exploits in the OS interfaces typically the system-call interface.
Virus	Malware that attaches itself to running programs and spreads itself through users' interactions with various systems.
Polymorphic Virus	A virus that, when replicating to attach to a new target, alters its payload to evade detection, i.e. takes on a di_erent shape but performs the same function.

Metamorphic Virus	A virus that, when replicating to attach to a new target, alters both the payload and functionality, including the framework for generating future changes
Trojan	Malware that masquerades as non-malware and acts maliciously once installed (opening backdoors, interfering with system behavior, etc).
AdWare	Malware that forces the user to deal with unwanted advertisements.
SpyWare	Malware that secretly observes and reports on users computer usage and personal information accessible therein.
Botnet	Malware that employs a user's computer as a member of a network of infected computers controlled by a central malicious agency.
Rootkit	Malware that hides its existence from other applications and users. Often used to mask the activity of other malicious software.

4. Machine Learning

Machine learning is a way of training algorithms to increase our understanding of a certain set of data. More specifically, machine learning attempts to develop algorithms from evaluating a set of training examples. It can be used to predict the outcome of new data based on previously analysed data, or to find patterns of similarity in a data set. It can also be applied to tasks where computers are to learn a certain type of behaviour based on some empirical data obtained during a training phase. One of the strengths of machine learning lies in the ability to perform tasks without explicitly programming an algorithm.

In machine learning, classifiers are able to examine data items to determine to which of N groups (classes) each item belongs. Often, classification algorithms will produce a vector of probabilities which represent the likelihoods of the data item belonging to each class. In the case of malware detection, we can simply define two classes: malware and non-malware. As a result, the output from each of classifiers will be two probabilities representing the likelihood of the data item being malware.

4.1. Unsupervised Machine Learning

Unsupervised learning studies how systems can learn to represent particular input patterns in a way that reflects the statistical structure of the overall collection of input patterns. There are no explicit target outputs or environmental evaluations associated with each input; rather the unsupervised learner brings to bear prior biases as to what aspects of the structure of the input should be captured in the output. We uses K-means clustering algorithm as unsupervised machine learning approach in our proposed framework.

4.1.1. K-Means Clustering

The features extracted from android applications are collected into the signature database and divided into training data and test data. Then, the proposed framework uses standard machine learning techniques to identify types of the android malware applications. We choose K-means clustering (i) it is data driven method relatively few assumptions on the distributions of the underlying data and (ii) it guarantees at least a local minimum of the criterion function, thereby accelerating the convergence of clusters on large datasets.

First stage: clustering is performed on training instances to obtain k disjoint clusters. Each cluster represents a region of similar instances in terms of Euclidean distances between the instances and their cluster centroids.

Second stage: K-means method is cascaded with classifiers by using the instances in each K-means cluster.

4.2. Supervised Machine Learning

Supervised machine learning is the search for algorithms that reason from externally supplied instances to produce general hypotheses, which then make predictions about future instances. In other words, the goal of supervised learning is to build a concise model of the distribution of class labels in terms of predictor features. The resulting classifier is then used to assign class labels to the testing instances where the values of the predictor features are known, but the value of the class label is unknown.

4.2.1. Classifiers

There are a large number of classifiers we could use. Classifiers broadly break down into two classes: linear and non-linear. Linear algorithms attempt to separate n-dimensional data points by a hyper-plane points on one side of the plane are of class X and points on the other side of class Y. Non-linear classifiers, however, have no such restrictions; any operation to derive a classification can be applied. Unfortunately, this means that the amount of computation to classify a data point can be very high. Here we briefly describe the algorithms we implement:

Specifically, the analysis procedure takes into account and cross-evaluates three supervised machine learning algorithms, i.e. Bayesian Network, RBF Network and Random Forest. A Bayesian network, also called a belief network model, is an annotated directed graph that encodes the probabilistic relationships among variables of interest. A Bayesian network classifier is a statistical classification eager method [15] that may be used as a classifier that gives the posterior probability distribution of the class node given the values of other features.

On the other hand, RBF is a type of ANN that consists of an input layer, a hidden layer and an output layer. Specifically, RBF is a single hidden layer feed-forward network and has a static Gaussian function as the nonlinearity for the hidden layer processing elements [16].

Finally, Random Forest is well-respected amongst the statistics and machine learning communities as a versatile eager method that produces accurate classifiers for many types of data [17]. Random forests (RF) are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlations between them. Using a random selection of features to split each node yields error rates that compare favourably to Adaboost, and are more robust with respect to noise. A common method for comparing supervised ML algorithms is to perform statistical comparisons of the accuracies of trained classifiers on specific datasets.

5. PROPOSED FRAMEWORK

In this paper, we propose a machine learning based identification framework for different types of android malicious applications. This framework is based on feature extraction as a first phase, dataset generation as a second phase, classification of this new dataset which is generated by second phase as third phase, and finally evaluating the performance of this proposed model in terms of accuracy, precision and recall. The proposed framework for android malware classification is shown in Figure 1.

5.1. Feature Extraction

Firstly, android application files (including malicious applications) are collected by downloading from the websites of android application market such as Google Play. Then, these downloaded application files are decompressed into the contents of android application files: classes.dex, AndroidManifest.xml, resources.rsc. Android application files are mainly focused on only two components: classes.dex and Androidmanifest.xml. Android application produces permission requests

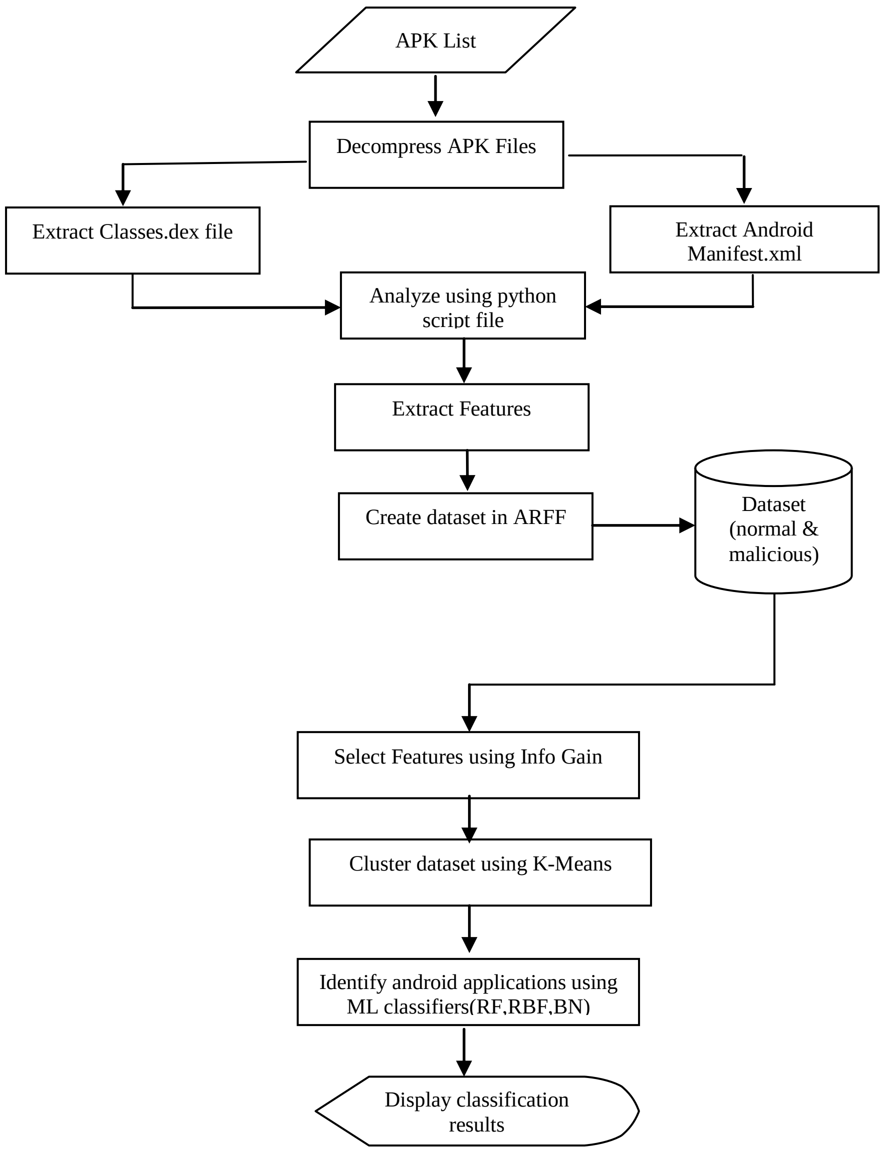
APK List
Decompress APK Files
Extract Classes.dex file
Extract Android Manifest.xml
Analyze using python script file
Extract Features
Create dataset in ARFF
Dataset (normal & malicious)
Select Features using Info Gain
Cluster dataset using K-Means
Identify android applications using ML classifiers(RF,RBF,BN)
Display classification results

Figure 1. Proposed Framework for Identification of Android Malicious Applications

at the installation time to confirm the allowance of the users. Application authors were developing their applications with appropriate permissions. However, some authors purposely hide the permissions they use in the application leading to application vulnerability. Most of the permission features are included in AndroidManifest.xml and classes.dex files. This proposed framework will extract these permission-based features from these important files by using python script file.

5.2. Dataset Generation

The permissions extracted from the first stage are used to create android malware dataset. This dataset is developed in ARFF (Attribute Relation File Format). This file consists of two parts. The first part contains the set of attributes and the values of these attributes are described in the second part.

6. EXPERIMENTS

We create five datasets with 100, 200, 300, 500, 1000 android applications. Firstly, we extracted the necessary features from android applications to identify types of malicious applications (Trojan, Infostel, Permsms, RootExploit and normal). Then, we built dataset (.arff) file from the extracted features. We use these five datasets to identify types of android malicious applications by machine learning approaches. Table 2 shows the details of five datasets used in the proposed framework and Table 3 shows the experimental results of different machine learning approaches by using these datasets.

Table 2. Datasets

Dataset Name	**Number of Samples**	**Number of Features**
Dataset #1	100	110
Dataset #2	200	110
Dataset #3	300	138
Dataset #4	500	190
Dataset #5	1000	212

Table 3. Experimental Results

Dataset Name	TP Rate	FP Rate	Precision	Recall	ROC Area	Correctly Classified Instances(%)	Incorrectly Classified Instances(%)
Dataset#1	0.856	0.137	0.867	0.856	0.921	85.57%	14.43%
Dataset #2	0.907	0.086	0.916	0.907	0.87	90.72%	9.27%
Dataset #3	0.817	0.19	0.828	0.817	0.912	81.68%	18.32%
Dataset #4	0.916	0.084	0.916	0.916	0.969	91.58%	8.42%
Dataset#5	0.973	0.027	0.975	0.986	0.964	97.43%	2.57%

7. Conclusions and future work

In this work, we have presented a machine learning-based system for the identification of malicious Android applications. As the vast majority of mobile malware targets the Android platform, this work focuses on Android malware identification. Our large scale experiments show that the classifiers are able to identify over 90% accuracy rate and few false alarms. By combining results from various classifiers, it can be a quick filter to identify more suspicious applications. Although the performance numbers are perfect, permission-based identification of malicious applications can be further improved. We believe that permission-based classifications can be a good auxiliary to detect malicious applications. However, the method presented can be adapted to other platforms with minor changes.

Acknowledgements

I would like to thank my supervisor, Dr. Win Zaw, Associate Professor and Head of Department of Information Technology, for his excellent guidance, where the initial scope of the thesis was defined, and throughout the process of writing this thesis.

References

[1] Schultz, M., Eskin, E., Zadok, F., Stolfo, S.: Data mining methods for detection of new malicious executables. In: Proceedings of the 2001 IEEE Symposium on Security and Privacy, 2001. S&P, IEEE (2001) 38-49

[2] Santos, I., Devesa, J., Brezo, F., Nieves, J., Bringas, P.: Opem: A static-dynamic approach for machine-learning-based malware detection. In: International Joint Conference CISIS12-ICEUTE12-SOCO12 Special Sessions. Herrero, .; Snel, V.; Abraham, A.; Zelinka, I.; Baruque, B.; Quintin, H.; Calvo, J.L.; Sedano, J.; Corchado, E. (Eds.). Advances in Intelligent Systems and Computing. Volume 189. (2012) 97-108

[3] Santos, I., Nieves, J., Bringas, P.G.: Semi-supervised learning for unknown malware detection. In: Proceedings of the 4th International Symposium on Distributed Computing and Artificial Intelligence (DCAI). 9th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS). (2011) 415-422

[4] Santos, I., Laorden, C., Bringas, P.G.: Collective classi_cation for unknown malware detection. In: Proceedings of the 6th International Conference on Security and Cryptography (SECRYPT). (2011) 251-256

[5] Santos, I., Brezo, F., Ugarte-Pedrero, X., Bringas, P.G.: Opcode sequences as representation of executables for data-mining-based unknown malware detection. Information Sciences ??(??) ?? in press, DOI:10.1016/j.ins.2011.08.020.

[6] Rieck, K., Holz, T., Willems, C., D□ussel, P., Laskov, P.: Learning and classification of malware behavior. Detection of Intrusions and Malware, and Vulnerability Assessment (2008) 108-125

[7] Tian, R., Batten, L., Islam, R., Versteeg, S.: An automated classi_cation system based on the strings of trojan and virus families. In: Proceedings of the 4th International Conference on Malicious and Unwanted Software (MALWARE). (2009) 23-30

[8] Shabtai, A., Fledel, Y., Elovici, Y.: Automated static code analysis for classifying android applications using machine learning. In: Proceedings of the International Conference on Computational Intelligence and Security (CIS). (2010) 329-333

[9] Burguera, I., Zurutuza, U., Nadjm-Tehrani, S.: Crowdroid: behavior-based malware detection system for android. In: Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, ACM (2011) 15-26

[10] Blasing, T., Batyuk, L., Schmidt, A., Camtepe, S., Albayrak, S.: An android application sandbox system for suspicious software detection. In: Proceedings of the 5th International Conference on Malicious and Unwanted Software (MALWARE). (2010) 55-62

[11] Shabtai, A., Elovici, Y.: Applying behavioral detection on android-based devices. In: Proceedings of the 3rd International Conference on Mobile Wireless Middleware, Operating Systems, and Applications: Mobilware. Volume 48., Springer Verlag (2010) 235

[12] Oberheide, J., Miller, J.: Dissecting the android bouncer. In: SUMECON 2012. (2012) http://jon.oberheide.org/files/summercon12-bouncer.pdf

[13] Chun-Ying Huang, Yi-Ting Tsai, and Chung-Han Hsu "Performance Evaluation on Permission-Based Detection for Android Malware"

[14] Juniper Networks. Juniper networks third annual mobile threats report, 2013.
[15] Heckerman D. A Tutorial on Learning with Bayesian Networks, Technical report 95-06, Microsoft Research Advanced Technology Division Microsoft Corporation, Redmond, USA, November. 1995.
[16] Neuro Dimension. Radial Basis Function. Available at: http://www.nd.com/models/rbf.htm
[17] Breiman L. (2001), Random Forests, Machine Learning 2001; 45(1): 5-32.