Detection of Android Malware Applications By Using Machine Learning Approaches

Zarni Aung
Faculty of ICT, UT-YCC
Pyin Oo Lwin, Myanmar
zarniaung.utycc@gmail.com

Win Zaw
Department of IT, TU(Than Lyin)
Than Lyin, Myanmar
winzaw@gmail.com

Abstract— **Mobile applications market has expanded at a very massive rate. Amongst mobile phones, android-based devices have received incredible adoption. The reason of the increased demand for android phones is the huge availability of applications that can be downloaded and installed easily on mobile phones. Android applications request the permissions allowance at installation. It is difficult for general users to differentiate between the set of permissions which are potentially harmful and those which are not. This weakness has become the chance for malware writers to infect the mobile devices. Therefore, we propose machine learning-based android malware detection framework to detect android malware applications in this paper. This framework extracts automatically various permission-based features from the android applications by using feature extraction tool written python script file, and create a dataset (in arff format) using these extracted features. Then, K-means clustering algorithm and three decision tree classifiers (J48, RF, CART) are used to detect android malware applications with better accuracy results over 90%.**

***Keywords*—android malware, machine learning, mobile applications, permissions**

I. INTRODUCTION

Mobile phones have evolved from simple into sophisticated yet compact minicomputers. Mobile devices become susceptible to various new threats such as viruses, Trojan horses and worms, all of which are well-known from the desktop computers [3]. Android is currently the most popular mobile phone operating system and users feel their private information at threat, facing a rapidly increasing number of malware for Android which significantly exceeds that of other platforms [1]. In 2011, malware attacks increased by 155 percent across all platforms [2]: in particular, Android is the platform with the highest malware growth rate by the end of 2011. Despite the rapid growth of the Android platform, there are already well-documented cases of Android malware, such as DroidDream, which was discovered in over 50 applications on the official Android market in March 2011 [4].

Android malware applications can be detected by using machine learning approaches. However, the problem of using machine learning approaches to detect malware applications presents two main challenges: first, given an android application, we must extract some sort of feature representation of the application; second, we have a data set that includes normal and different types of malicious applications so that we must choose machine learning classifier that can be trained on multi-classes. To address the first problem, we extract android permission features from the application files. To address the second problem, we use decision tree classifiers (RF, CART, J48) which we train using benign and malicious applications.

This paper is organized as follows: in Section II, we discuss related work; in Section III, we present the mobile malwares according to mobile operating systems or platforms. Android applications and their components are described in Section IV, we proposed android malware detection framework by using an automatic feature extraction tool in Section V, we present machine learning approaches to detect android malware applications in Section VI, we discuss our experimental results in Section VII; finally in Section VIII, we conclude about the proposed system.

II. RELATED WORK

Android malware applications have been rapidly rising and there are several approaches to detect these malware applications. The approaches [15], [16] [17] focus on using machine learning and data mining approaches for malware detection. In [17], Tesauro *et al.* train a neural network to detect boot sector viruses, based on byte string trigrams. Schultz *et al* [16] compare three machine learning algorithms trained on three features: DLL and system calls made by the program, strings found in the program binary, and a raw hexadecimal representation of the binary. In [15], Kolter and Maloof train several machine learning algorithms on byte string n-grams. [5] presents a machine learning based system for the detection of malware on Android devices but, this system extracts a number of features and trains a One-Class Support Vector Machine in an offline (off-device) manner in order to leverage the higher computing power of a server or cluster of servers.

III. MOBILE MAWARES

Mobile malwares have been rapidly increased depending on the platforms. Cabir is the first malware that aimed to infect mobile devices running Symbian operating system through their Bluetooth connection. DroidKungFu and GinMaster attempt to steal private information much like their desktop counter parts. FakePlayer Trojan is often considered the first malware aimed at Android. Since the first malicious Android application was discovered in 2010, the number of malicious applications has been consistently rising. The population rate of android malware applications is the highest as shown in the following pie chart.

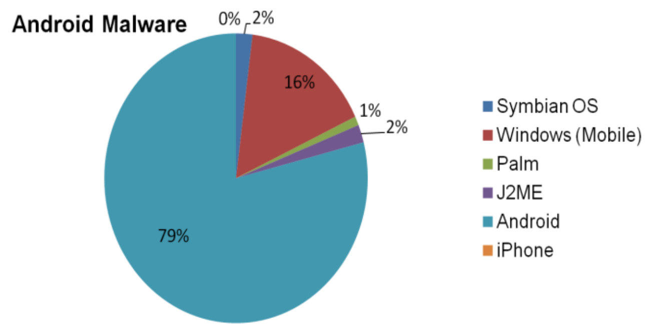


Figure 1: Mobile Malware Per Platform By 2010

A. *Attacks on Smartphones*

For the sake of the study, the attacks are distinguished in two types as follows: 1) Attacks related to permission-model (data-stealing attacks) and 2) Attacks not related to permission-model.

1) Attacks related to permission-model: In such type of attacks, an attacker made the third party application and uploaded it on the Google Play. Android users download this application and install it on their smartphones. The application asks different permissions at the time of installation. As the Android Permission Model is coarse-grained, user is bound to give access to all the requested permissions in order to successfully install the application [6]. Now, while the user uses the application, in the back-end the application can misuse the permission granted to it to steal the private data of the user. Such attack exploits the vulnerability in the design of Android Permission Model.

2) Attacks not related to permission-model: These types of attacks consist of performance degradation attacks, denial of service attacks and other such attacks. Various anti-virus and other attack detection mechanisms have been proposed to get rid of such attacks maintaining the integrity of the specifications.

B. *Android Malwares*

There are several malwares facing the Android operating system. Generally speaking, all Android malwares are Trojans. The attack vectors used by viruses and worms are largely unavailable to the malware developers. Then, one of the most common types of malicious applications for the Android platform is spyware that comes in two favors; commercial and malicious. Commercial spywares are applications installed on the user's handset manually by another person specifically to spy on the user, while malicious spyware operates in a similar fashion as its desktop counterpart; covertly stealing data and transmitting it to a third party. Like most Trojans, the malicious application pretends to be normal until it is installed on the user's device. When installed, it attempts to use one or more root exploits to gain root access to the device. Botnet is a network of compromised devices, usually computers, which an attacker can use for his own purposes; often to steal sensitive data or as part of a denial of service attack.

IV. Android Application Permissions

Permissions are requested by an application during the install process to grant access to various features and functionalities on a device. Currently there are 124 unique permissions which are categorized into 11 top level groups [6]. These permissions range from services that cost your money to access of your personal information. The permission model is a good security feature because it forces developers to explicitly ask for the specific functionalities needed for their application to function. These permissions are displayed before any application is installed and can also be viewed post installation. The downfall is that users cannot be expected to understand all 124 permissions or the associated risks with a few specific permissions. Not having the comprehensive knowledge of the applications the developer will have, it is also impossible for users to know which permissions are actually needed by an application. When installing an application, the permissions needed are displayed; but users do not have the ability to decline certain permissions thereby choosing which functionality an application can access versus which are forbidden. With these nice security features, why is it that Android has a malware problem but iOS does not? To better understand this we need to fully understand the application (APK) format which is unique to Android devices.

A. *Components of Android Applications*

For malware analysis, it is important to understand the different components that can be found in Android applications [11]:

1) Activities: They represent a single screen with a user interface. Usually this kind of component is implemented by the original applications that are being infected by malicious code.

2) Services: The principal characteristic of this component is that it runs in the background. For this reason, it is commonly used by malicious code that is being packed into legitimate applications.

3) Content providers: The purpose of this component is to provide the ability to share data between applications. This kind of component is not commonly used by malware authors.

4) Broadcast receivers: This component "responds to system-wide broadcast announcements." It is usually used by malicious code to trigger specific functionality when an event occurs in the device(for example, the screen is turned off, the device has a wireless Internet connection, or a call is being received).

5) Finally, the last layer of the Android architecture is where the final application is compiled and ready to be installed and executed in the system. The format of this file is called an Android Package (.apk), which is, in fact, a compressed file (just like a jar file) that contains, at least, the following components:

a) Manifest: The purpose of this file is to declare all the components, requirements, and permissions required by the application to be executed in the system. The file is written using a standard XML structure and must be present at the root of the application project directory.

b) Classes.dex: Inside this file there is Dalvik bytecode that must be decompiled/disassembled in order to perform a static analysis of the Android application.

c) Resources: It contains all the information related to the resources used by the application (images,audio files, menus, and more)

V. Proposed Framework

In this paper, we propose a malicious application detection framework on android by using machine learning approaches. This framework uses a feature extraction tool written by python script file to extract android permission features. The proposed framework is shown in Figure 2. In this framework, the android applications on android market are firstly downloaded and decompressed into the contents of android applications. The AndroidManifest.xml and classes.dex files are only selected because these two files contain the necessary permissions features. Then, the feature extraction tool extracts the features that based on permissions from these two files. This helps to increase a detection rate and decrease a false negative rate on the static detection.

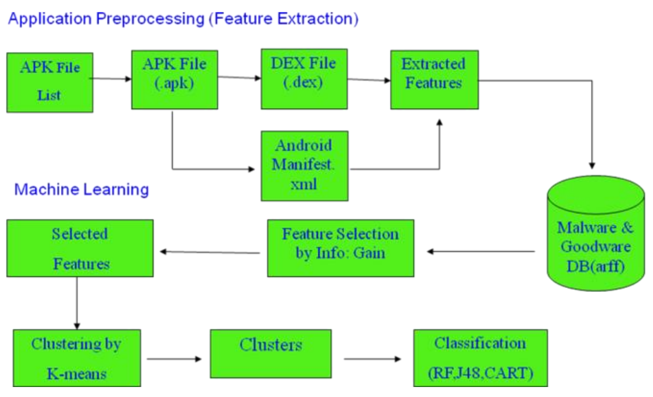
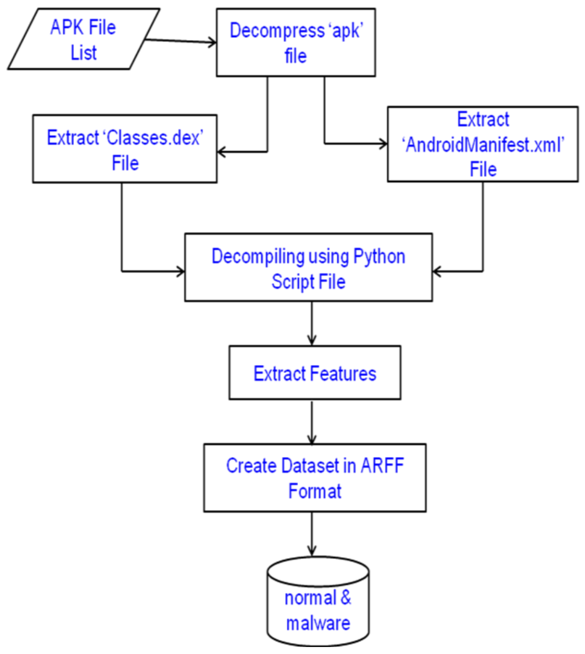


Figure 2: Android Malware Detection Framework

A. *Feature Extraction Tool*

To extract features before machine learning, we use an automatic feature extraction tool written by python script. Figure 3 is the overall process of automatic feature extraction. As an input, this tool takes an '.apk' file as android application package. The tool decompresses the '.apk' file because the '.apk' file is a compressed bundle of files. When the '.apk' file decompressed, this file is split into three main contents such as 'AndroidManifest.xml' file, 'Classes.dex' file and 'res' folder. The 'AndroidManifest.xml' file is an XML file including the set of permissions used in application. The 'Classes.dex' file is application source code which holds the complete bytecode to be interpreted by Dalvik Virtual Machine [12].The 'res' folder is consist of files defining the layout, language, etc. The automatic feature extraction tool extracts the 'AndroidManifest.xml' file and the 'Classes.dex' file related to permissions. Then, the tool decompiles the 'Classes.dex' file and 'AndroidManifest.xml' using python script file. Finally, using our program written by python script, the automatic feature extraction tool creates the dataset that contains the permission features and their relevant data.



APK File List
Decompress 'apk' file
Extract 'Classes.dex' File
Extract 'AndroidManifest.xml' File
Decompiling using Python Script File
Extract Features
Create Dataset in ARFF Format
normal & malware

Figure 3: Automatic Feature Extraction Tool

B. *Feature Selection*

In Machine Learning applications, a large number of extracted features, some of which redundant or irrelevant, present several problems such as—misleading the learning algorithm, over-fitting, reducing generality, and increasing model complexity and run-time. These adverse effects are even more crucial when applying Machine Learning methods. Applying fine feature selection in a preparatory stage enabled to use our malware detector more efficiently, with a faster detection cycle. Nevertheless, reducing the amount of features should be performed while preserving a high level of accuracy. In this section we select the k best features from the extracted features of android application package files by using feature selection method: Information Gain. This method depends on entropy of the attributes and it selects the largest value of gain as the best feature.

VI. Machine Learning Approaches

Machine-learning is an active research area within Artificial Intelligence (AI) that focuses on the design and development of new algorithms that allow computers to reason and decide based on data (i.e., computer learning) [7]. Machine learning algorithms can commonly be divided into three different types: supervised learning, unsupervised learning and semi-supervised learning. For supervised algorithms, the training dataset must be labeled (e.g., the category of an application) [8]. Unsupervised learning algorithms try to determine how data are organised into different groups named clusters. Therefore, data do not need to be labeled [9]. Finally, semi-supervised machine-learning algorithms use a mixture of both labeled and unlabelled data in order to build models, improving the accuracy of unsupervised methods [10]. Because android applications can be properly labeled, we use supervised and unsupervised machine learning methods for detection of android malware applications.

A. K-Means Clustering

K-means clustering is one of the most common or popular techniques. Each cluster is associated with a centroid (center point) – this is often the mean – it is the cluster *prototype*. Each point is assigned to the cluster with the closest centroid. The number of clusters, k, must be specified ahead of time.

Algorithm

1. Select K points as the initial centroids.
2. Repeat
3. From K clusters by assigning all points to the closest centroid.
4. Recompute the centroid of each cluster
5. Until The centroids don't change

B. Decision Tree Classifiers

Decision tree classifiers are tree-based classifiers for instances represented as feature-vectors. They recursively partitions a data set of records and use a depth-first greedy approach or breadth-first approach. Nodes are used for test features, there is one branch for each value of the feature, and leaves specify the category until all the data items belong to a particular class. The advantages are:

1. Robustness to noise, low computational cost
2. For generating the model, and ability to deal with redundant attributes.
3. Use different variants of impurity measures, like, information gain, gain ratio, gini-index and distance-based measures, to select an input attribute to be associated with an internal node.

1) Classification and Regression Trees(CART)

CART partitions the feature space into a set of rectangles and fit a simple model in each one. Then, it constructs binary tree structured classifiers by repeated splits of subsets (nodes) of the measurement space X into two descendant subsets (starting with X itself). This method assigns a class label for each terminal subset and the resulting partition of X corresponds to the classifier. It uses gini index splitting measure in selecting the splitting attribute. This classifier makes pruning by using a portion of the training data set and uses both numeric and categorical attributes for building the decision tree and has in-built features that deal with missing attributes.

2) Random Forests(RF)

It is an ensemble method specifically designed for decision tree classifiers and uses recursive partitioning to generate many trees and then combine the result. This method grows many classification trees and ensemble of unpruned decision trees. Each tree is independently constructed using a bootstrap sample of the data. Forest chooses the classification having the most votes (over all the trees in the forest).

3) C4.5(J48)

C4.5 (know as a J48) is a system that constructs classifiers. It uses simple depth-first construction and the information gain of the data attribute to sort the data. These classifiers are one of the commonly used tools in data mining. Such systems take as input a collection of cases, each belonging to one of a small number of classes and described by its values for a fixed set of attributes. With that, a classifier accurately predicts the class to which a new case belongs.

VII. Experimental Results

To evaluate the proposed framework, we collected 1000 including normal applications from android market and malicious applications from the internet site [13]. Then, features were extracted and dataset was created by an automatic feature extraction tool. The machine learning was done by WEKA which is a machine learning tool [14]. We used K-means clustering algorithm to cluster the android applications from the dataset with selected features. Decision tree classifiers (CART, J48, RF) were used to classify android malware applications with better accuracy results and k-folds cross validation scheme was used to train and test the data set. The experimental results of our proposed framework according to sample size are shown in Figure 4.

VIII. Conclusion And Future Work

Mobile phones have become the central computation and communication devices, and they are starting to replace traditional computers because of their superior mobility and nearly Internet access. At the same time, Android is getting a lot of attention, and its popularity is growing exponentially each year. Because of the popularity and ubiquity of mobile phones, malware authors are starting to develop new threats for this platform that are being actively distributed via what is supposed to be a trusted source: the official Android Market. Therefore, a framework for detection of android malware applications using machine-learning techniques has been proposed. As the future work, we will collect android malware applications systematically and we will use the other features besides permission-based features to detect android malware applications.

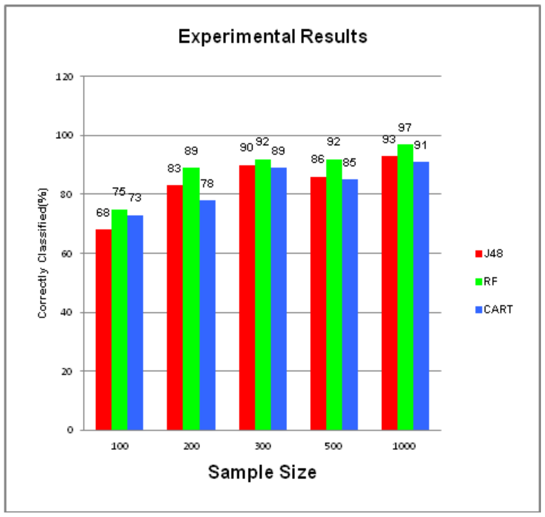


Figure 4: Accuracy Measures Vs Sample Size

ACKNOWLEDGMENT

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

REFERENCES

[1] Rafael Fedler, Julian Schutte, Marcel Kulicke, "On The Effectiveness of Malware Protection on Android:An Evaluation of Android Antivirus Apps", Technical Report Antivirus Test (April 2013)

[2] Juniper Networks: 2011 Mobile Threats Report (February 2012)

[3] Asaf Shabtai, "Malware Detection on Mobile Devices", on 11th International Conference on Mobile Data Management

[4] Lookout Mobile Security. Security alert: Droiddream malware found in official android market. http://blog.mylookout.com/blog/2011/03/01/security-alert-malware-found-in-official-android-market-droiddream/.

[5] Justin Sahs and Latifur Khan, "A Machine Learning Approach to Android Malware Detection", European Intelligence and Security Informatics Conference, 2012

[6] Nitin B. Satpal, "Enhancing Permission Model of Android", (June 2013)

[7] C. Bishop, Pattern recognition and machine learning. Springer New York., 2006.

[8] S. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," in Proceeding of the 2007 conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies, 2007, pp. 3–24.

[9] S. Kotsiantis and P. Pintelas, "Recent advances in clustering: A brief survey," WSEAS Transactions on Information Science and Applications, vol. 1, no. 1, pp. 73–81, 2004.

[10] O. Chapelle, B. Sch¨olkopf, and A. Zien, Semi-supervised learning. MIT Press, 2006.

[11] Carlos A. Castillo, Android Malware Past, Present, and Future, Mobile Security Working Group, McAfee

[12] http://en.wikipedia.org/wiki/Dalvik_VM

[13] http://contagiominidump.blogspot.kr/

[14] http://www.cs.waikato.ac.nz/ml/weka/

[15] J. Zico Kolter and Marcus A. Maloof. Learning to detect and classify malicious executables in the wild. *J. Mach. Learn. Res.*, 7:2721–2744, December 2006.

[16] Matthew G. Schultz, Eleazar Eskin, Erez Zadok, and Salvatore J. Stolfo. Data mining methods for detection of new malicious executables. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, SP '01, pages 38–, Washington, DC, USA, 2001. IEEE Computer Society.

[17] G.J. Tesauro, J.O. Kephart, and G.B. Sorkin. Neural networks for computer virus recognition. *IEEE Expert*, 11(4):5 –6, aug 1996.