IJCSN - International Journal of Computer Science and Network, Volume 8, Issue 3, June 2019
ISSN (Online) : 2277-5420
www.IJCSN.org
Impact Factor: 1.5

Machine Learning Based Android Malware Detection Using Permission and Function Call

[1] Emi Johnson; [2] Deepa Rajan S; [3] Radhakrishnan B

[1] CSE Department, BMCE,
APJ Abdul Kalam Technological University, Sasthamcotta, Kerala, India

[2] CSE Department, BMCE,
APJ Abdul Kalam Technological University, Sasthamcotta, Kerala, India

[3] CSE Department, BMCE,
APJ Abdul Kalam Technological University, Sasthamcotta, Kerala, India

Abstract- Over the last few years smart phones have brought peoples lives to a new high level. With this rapid development of smart phones, android a Linux based open source mobile operating systems popularity will also increase . Android is mainly designed for touch screen devices such as tablets, smart phones, etc. Also, it allows the use of third -party applications. Due to the widespread use of android operating system, it has become one of the favourite targets of cyber criminals. So an effective and efficient malware detection system is very important. This paper proposes a machine learning based technique that is capable of finding malware and benign applications. Here first we reduce the total number of permission of an android application using reduction techniques . This technique consist of three steps.1.Scoring permission using their negative values 2.Scoring permission using support 3.Permission mining using association rule. Using this reduction technique we find the most significant 22 permission . After reduction technique we analyse the function calls to permission controlled function. As a result, this technique provide a more efficient and effective result to determine whether an application is malware or benign.

Keywords *-Android App, Android Permission, Malware, Benign.*

1. Introduction

Now a days the popularity of smart phones are increasing. These small devices help the users in many ways and everywhere. For example, for browsing the internet, playing games etc. It is vital to install applications on these phones in order to take all the advantages that the device offers. Android is the most popular operating system used in all these smart devices. Android is a Linux-based open source operating system developed by google. Android is mainly designed for touch screen devices such as smart phones, tablets etc. Applications in android are written using Android software development kit (SDK). Android allows the use of third -party applications, by downloading and installing application package (APK) file or by downloading from application store of android. Google play store is the store for android applications. There are nearly 3 million apps available for download in google play store.

There are several application types in Android: native applications (developed with the Android SDK), web applications (developed mostly with HTML, JavaScript ad CSS) and widgets (simple applications for the Android desktop, which are developed in a similar way to web applications). Permissions are the most recognisable security feature in Android. User must accept them in order to install the application. We performed a study of the different permissions of the applications, in order to determine their suitability for malware detection. These permissions are evaluated when installing the app, and must be approved by the user. We used the Android Asset Packaging Tool (aapt) to extract and decrypt the data from the AndroidManifest.xml file, provided by the Android SDK. Android applications is distributed and installed using Android application package(APK). APK file contain bytecode, configuration, precompiled library and resource files. In order to perform analysis on android applications unpack these package and need to extract Android Manifest and Dex byte code files[1].

There are so many reasons why android become the frequent victim of malware. Mainly android has a loosely controlled ecosystem. As a result many users and developers choose to publish and download app from third party application stores. The second thing is that many manufactures are licensed to introduce wide variety of smart phones ,tablets etc which make a large and complex family of android versions and hardware configurations. The lack of evenness in operating systems and hardware make the support and maintenance inefficient and insufficient. Third one is that, most of the android apps are developed using java, which is sometimes called as semi compiled language. Here first part of compilation is performed by the programmer and the second part of compilation is carried out on the client machine. Due to this reason applications are relatively

easy to decompile and the vulnerabilities in Dalvik Virtual Machine may also exploited by some malware. Malware detection system has been used to determine whether a program or application has malicious intent or not. Malware comes in various forms and categories. They are mainly classified according to their propagation method and their performed actions on the infected machines.

These are some of the common types of malware.

- Virus: It is malicious app or program that transfer from one program to another or from one machine to another by inserting their code.
- Worm: A computer worm is a self-replicating program which spread from one machine to another by transferring a copy of itself via network without user authentication.
- Trojan horse: It is a type of computer software that typically destroy data or attempt to extract confidential information including financial data ,passwords etc
- Spyware: Is any software installed on a machine without users awareness.it generally enters a system when a trial or free software is downloaded and installed on the system .

The primary techniques of detecting malware applications on android are:

Static Analysis: The Static Analysis screen's part of the application without really executing them. This analysis incorporates Signature-based, Permission-based and Component-based analysis. The Signature-based strategy draws features and makes distinctive signs to identify specific malware. Hence, it falls short to recognize the variation or unidentified malware. The Permission-based strategy recognizes permission requests to distinguish malware. The Component-based techniques decompile the APP to draw and inspect the definition and byte code connections of significant components (i.e. activities, services, etc.), to identify the exposures. The principal drawbacks of static analysis are the lack of real execution paths and suitable execution conditions

Dynamic Analysis: This technique includes the execution of the application on either a virtual machine or a physical device. This analysis results in a less abstract perspective of application than static analysis. The code paths executed during runtime are a subset of every single accessible path. The principal objective of the analysis is to achieve high code inclusion, since every feasible event ought to be activated to watch any possible malicious behaviour

Hybrid Analysis: The hybrid analysis technique includes consolidating static and dynamic features gathered from examining the application and drawing data while the application is running, separately. Nevertheless, it would boost the accuracy of the identification. The principal drawback of hybrid analysis is that it consumes the Android system resources and takes a long time to perform the analysis.

The rest of this paper organized as follows a brief review of existing works on malware detection on android applications is given in section II. Our proposed method is explained in section III. The experimental results are discussed in section IV. Finally section V concludes the paper.

2. Related Work

Mengyu Qiao[1] proposed a novel machine learning approach to detect malware by mining the patterns of Permissions and API Function Calls acquired and used by Android Apps. Sanya Chaba[2] observe the system call log of each application, use the same for the construction of a dataset, and finally use this dataset to classify an unknown application as malicious or benign. Daniel Arp[3] propose DREBIN, a lightweight method for detection of Android malware that enables identifying malicious applications directly on the smartphone. DREBIN performs a broad static analysis, gathering as many features of an application as possible. These features are embedded in a joint vector space, such that typical patterns indicative for malware can be automatically identified. KhaledRia[4] propose Rough Droid, a floppy analysis technique that can discover Android malware applications directly on the smartphone. Rough Droid is based on seven feature sets($FS1$,$FS2$,...,FS 7)from the XML manifest file of an Android application ,plus three feature sets($FS8$,$FS9$,and$FS10$)from the Dexfile. Those feature sets pass through the Rough Set algorithm to elastically classify the Android application as either benign or malicious.

3.Proposed System

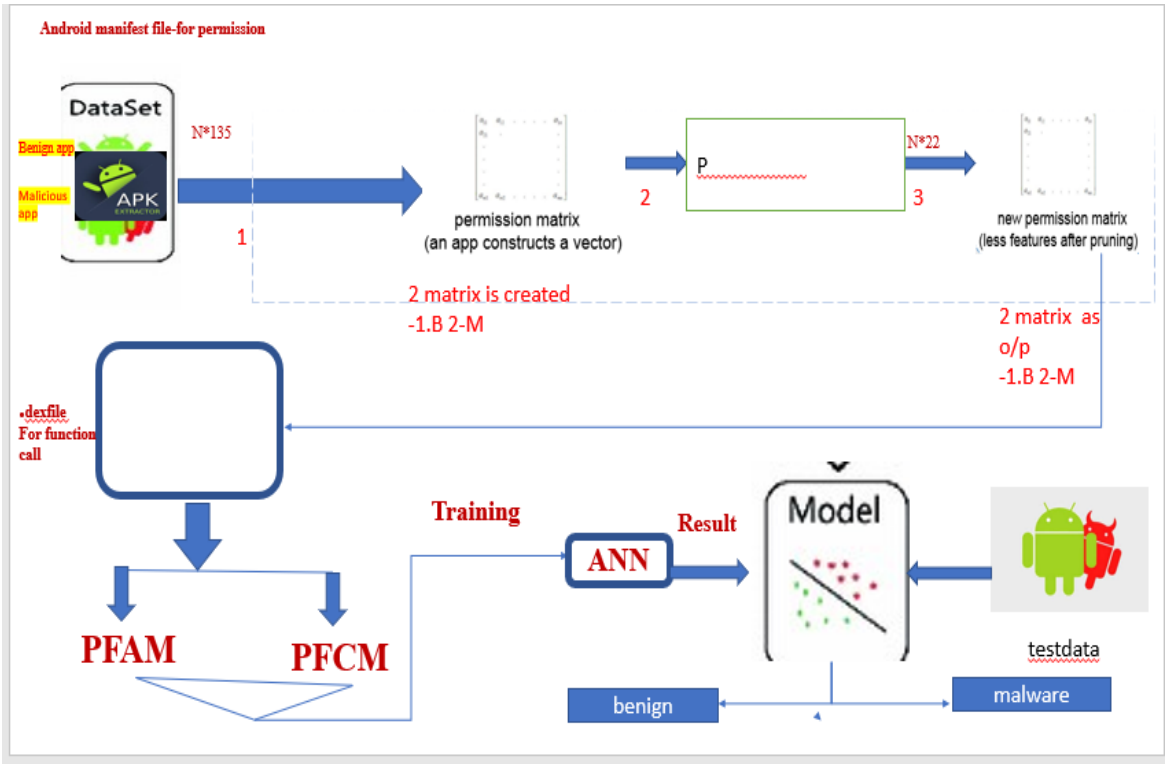

Fig1: Architecture of the proposed model

The goal of our system is to achieve high malware detection accuracy and efficiency while analysing the minimal number of permissions and finding functions used by each application. To achieve this goal, our system extracts permission uses from application packages, but instead of focusing on all the requested permissions, this method mainly focuses on permissions that can reliably improve the malware detection rate. This approach, in effect, eliminates the need to analyse permissions that have little or no significant influence on malware detection effectiveness. This system consist of 3 main parts for reduction technique
1.Scoring their permission using their negative values
2.Permission mining using association rules
3.Scoring permission using support.

1.Scoring their permission using their negative values:- Each permission describes a particular operation that an app is allowed to perform. For example, permission INTERNET indicates whether the app has access to the Internet. Different types of benign apps and malicious apps may request a variety of permissions corresponding to their operational needs. For malicious apps, we hypothesize that their needs may have common subsets, and we do not need to analyse all the permissions to build an effective malware detection system. As a result, on one hand, our focus is more on the permissions that create high risk attack surfaces and are frequently requested by malware samples. On the other hand, the permissions that are rarely requested by malware samples are also good indicators in differentiating between malicious and benign apps. Therefore, Our reduction procedure identifies both types of highly differentiable permissions so that we can use this information to classify malicious and benign apps. At the same time, we exclude permissions that are commonly used by both benign and malicious apps, as they introduce ambiguity in the malware detection process. For instance, permission INTERNET are frequently requested by both malware and benign apps, as almost all apps will request to access the Internet. Therefore, our approach prunes permission

INTERNET. To identify these two types of significant permissions, we design a permission ranking or scoring scheme to rank permissions based on how they are used by malicious and benign apps. Ranking is not a new concept. Prior works have also used generic permission ranking strategy such as mutual information to identify high risk permissions [5].

This scoring approach operates on two matrices, M and B. M represents a list of permissions used by malware samples and B represents a list of permissions used by benign apps. M_{ij} represents whether the j^{th} permission is

requested by the i^{th} malware sample, while '1' indicates yes, '0' indicates no. B_{ij} represents whether the j^{th} permission is requested by the i^{th} benign app sample. Before computing the support of permissions from matrices M and B, we first check their sizes.

Training over imbalanced dataset can lead to skewed models [6]. To balance the two matrices, we use Equation 1 to calculate the support of each permission in the larger dataset and then proportionally scales down the corresponding support to match that of the smaller dataset. In case that the number of rows of B is bigger than that of M, we have:

$$S_B(P_j) = \frac{\sum_i B_{ij}}{size(B_j)} * size(M_j) \text{-----(1)}$$

where Pj denotes the jth permission, and SB(Pj) represents the support of jth permission in matrix B. scoring can be implemented using equation (2) $R(P_j) = \frac{\sum_i M_{ij} - S_B(P_j)}{\sum_i M_{ij} + S_B(P_j)}$--------(2)

The scoring is used to perform ranking on our data set The result of $R(P_j)$ has a value ranging between [-1, 1]. If $R(P_j) = 1$, this means that permission P_j is only used in malicious dataset, which is a high risk permission. If $R(P_j) = -1$, this means that permission P_j is only used in benign dataset which is a low risk permission. If $R(P_j) = 0$, this means that P_j has very little impact on malware detection effectiveness. Since both -1 and 1 are important, we build two ranked lists: one list is generated in an ascending order based on the value of $R(P_j)$, and the other list is generated in a descending order.

2.Permission mining using association rule-Here we reduce permissions that always appear together in an app. For example, permission WRITE SMS and permission READ SMS are always used together. They also both belong to the Google's "dangerous" permission list. So here we use one which has a higher support, to its partner. In this example, we can remove permission WRITE SMS. Association rule mining has been used for discovering meaningful relations between variables in large databases. For instance, if event A and B always co-occur, it is highly likely that these two events are associated. In this paper, we only consider rules with high confidence. Apriori uses a breadth-first search strategy to count the support of itemset and uses a candidate generation process, which exploits the downward closure property of the support. Here, we only want to generate the association rules with high confidence even if the permissions have small support values.

3.Scoring permission using support-To further reduce the number of permissions, we turn our focus to the support of each permission. Typically, if the support of a permission is too low, it does not have much impact on malware detection performance. For instance, we find the permission BIND TEXT SERVICE only in benign apps. As a result, we may consider that any app that uses BIND TEXT SERVICE is benign.

After completing the reduction technique , unpack the apk file and extract dexfile for functions used by each application. Then we perform a mapping between reduced permission set and function call used by each application. From this set we can find the dangerous combination . A double filtering technique is actually done here for each application.

3.1 Multi Layer Perceptron

A multilayer perceptron (MLP) is a deep, artificial neural network. It is composed of more than one perceptron. They are composed of an input layer to receive the signal, an output layer that makes a decision or prediction about the input, and in between those two, an arbitrary number of hidden layers that are the true computational engine of the MLP. MLPs with one hidden layer are capable of approximating any continuous function . perceptron is a linear classifier; that is, it is an algorithm that classifies input by separating two categories with a straight line A perceptron produces a single output based on several real-valued inputs by forming a linear combination using its input weights (and sometimes passing the output through a nonlinear activation function.

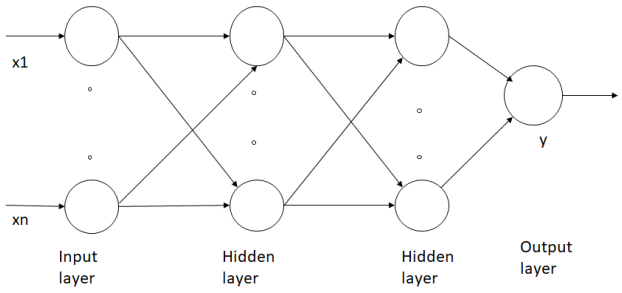

Fig2: The Multilayer Perceptron Model

4. Collected Dataset

We generate ten datasets by randomly choosing 5,494 benign apps from 310,926 benign apps, downloaded from Google play store in June 2013, to carry out cross validation.

The malicious apps are classified into 178 families, and the benign apps are grouped into a single family.

We select the number of benign apps to be the same as malicious apps to maintain balance during training, as the imbalanced dataset can result in skewed models [6].

5. Result and Discussion

The proposed malware detection approach incurs less processing times than those incurred by an approach that analyses the complete permission list. Additionally, smaller feature set can also reduce memory consumption. Based on the tested 25 machine learning algorithms, we also find that the machine learning methods based on MLP can produce better results. This experiment provides the evidence that the proposed approach can be used to effectively detect malware beyond the initial data set that we used for training and testing, which shows the potential applicability of SIGPID in practice.

6. Experimental Result and Analysis

The proposed malware detection approach incurs less processing times than those incurred by an approach that analyses the complete permission list. Additionally, smaller feature set can also reduce memory consumption. Based on the tested 25 machine learning algorithms, we also find that the machine learning methods based on MLP can produce better results. This experiment provides the evidence that the proposed approach can be used to effectively detect malware beyond the initial data set that we used for training and testing, which shows the potential applicability of SIGPID in practice.

Table 1: Result Analysis

ALGORITHM	ACCURACY
RF	91.05%
SVM	89.1%
DECISION TREE	90.02%
PROPOSED	94.6%

7. Conclusion

In this paper, we have shown that it is possible to reduce the number of permissions to be analysed for mobile malware detection, while maintaining high effectiveness and accuracy. To extract only significant permissionsthroughasystematic,3-level of reduction technique is used. We have developed a malware detection system based on permission and API calls. Our approach performs as well as or better than techniques that consider more permissions or all permissions. By using significant permission, we can improve performance a lot. Based on our dataset, which includes over 2,000 malwares, we only need to consider 22 out of 135 permissions to improve the runtime performance.

Reference

[1] Merging Permission and API features for Android Malware detection ;Mengyu Qiao
[2] Malware Detection Approach for Android systems Using System Call Logs; Sanya Chaba
[3] DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket; Daniel Arp
[4] Rough Droid: Operative Scheme for Functional Android Malware Detection; KhaledRiad
[5] W. Wang, X. Wang, D. Feng, J. Liu, Z. Han, and X.Zhang, "Exploring permission-inducedriskinandroidapplicationsformaliciousapplication detection," Information Forensics and Security

Author Profile

Emi Johnson is pursuing her M.Tech. degree on Computer Science and Engineering at APJ Abdul Kalam Technological University, Thiruvananthapuram. Her research interests are in Image processing, Data Mining and Data Security.

Deepa Rajan S is working as Assistant Professor in Computer Science and Engineering Department. She has 10 years experience in teaching. Her research interests focuses on Data Security, Image Processing.

Radhakrishnan B is working as the Head of CSE department. He has more than 14 years experience in teaching and has published papers on data mining and image processing. His research interests include image processing, data mining, image mining.