Malware Detection In Android Smart Phones Using Machine Learning

Sheikh Shahid Nazir
Department of Computer Science & Engineering
Desh Bhagat University
Punjab, India

Navneet Kaur Sandhu
Department of Computer Science & Engineering
Desh Bhagat University
Punjab, India

Abstract: Malware or Malicious Software is defined as software intended to misrepresent and interrupt the mobile or computer applications, gather important information and hence, carry out malicious operations These malicious operations consist of gaining access to secret information; stealthily steal this valuable information over the system, show undesirable advertisement, and spy on the activities of the users. Malicious software or malwares are programs that are created to hard ,interrupt or damage computers, networks and other resources associated with it, mostly the medium used to spread malwares networks. Malwares are always been a threat to digital world, the impact of the malwares become severe.

Mobile Phones are replaced by smart phones recurrently with most of the smart phones having Android applications running on them. Smart phones are used by the users for two types of functions. Firstly, the Android applications on the smart phones are integrated for individual use i.e. to obtain pictures, contacts, emails and other agendas. Secondly, the very same device is also subjugated to retrieve the IT infrastructure of an organization. The latter is implemented for policies like Bring Your Own Device. In the same scenario, there is an increase in the number of malwares attacking these devices The present mobile market, Android is the most well-known stand for smart phones, the market expansion rate is rising slowly but surely and it is now at 84.7%.With the speedy expansion of mobile computing technology, tablets and smart phones have evolved classy functions at minor costs. While, other platforms like iOS users permit to install applications only through iTunes, Android allows many open sources, such as Torrents, Google play store, Direct downloads, or Third-party markets, etc.(Handa, 2015) This sovereignty makes allocation of applications and bundling of malware an easy task for attackers (Xialoeiwang, 2015) who try to entice the users into running malicious code. Malicious payloads are used in repackaging popular apps (Hiranwal, 2013). Privacy breaches (e.g., GPS coordinates and access to address book), monetization through premium calls and SMS, other dangerous malicious attacks have become real threats.

***IndexTerms* — Android malware detection, malware detection techniques, Machine learning classifiers.**

I. Introduction

Smart phone market is expanding expanding consistently. However, the increase of usage of mobile phones caused a serious problems at the same time, malware targets the applications that are runs over the internet .malware creators are always come with new ideas. They develop malwares in such a way that they changed themselves from time to time that the cannot be detected easily. They always try to write a program that cannot be easily detected

The present mobile market, Android is the most well-known stand for smart phones, the market expansion rate is rising slowly but surely and it is now at 84.7%.With the speedy expansion of mobile computing technology, tablets and smart phones have evolved classy functions at minor costs. While, other platforms like iOS users permit to install applications only through iTunes.

Android allows many open sources, such as Torrents, Google play store, Direct downloads, or Third-party markets, etc.(Handa, 2015). Malware detectors are ued to detect these malwares and antivirus a canners are one of the way to detect them with progression of malware development techniques , malware detectors use a number of techniques to avoid the disastrous effects of the softwares . Due to the limitations of existing malware detection techniques , the machine learning and data mining are combined with existing detection methods to add the efficiency in the detection process

Android system separates privilege from advertising library to access sensitive information allowed for other permissions and internet information

II. LITERATURE REVIEW:

The static analysis techniques have been applied in various different fields in the previous research works:

Static analysis technique**(RaviKiranVerma, 2017)** use different machine learning algorithms such as Navies Bayes, Random forest, to detect Android malware and calculate the performance of all algorithms separately. Here they implemented a framework for classifying Android applications with the help of machine learning techniques to confirm whether it is malware or normal application. To use this model, they mined several features (permission) from many downloaded applications from the Android market. For validating, they collected 3258 samples of Android apps and those have to be extracted for each and every application, extract their features and have to train the models going to be evaluated with the help of classification accuracy and time taken for the model. As far as computational complexity is considered Naive Bayes classifier was found to be the fastest in classification of malware data set. This work is also compared with existed works in this area and the results obtained are superior regarding accuracy of classification. Feature reduction greatly improves the performance of the classifier.

Androtracker Android Malware detection **(Hyun Jae Kang, 2017**) is based on static analysis that uses serial number from the certificate issued by the certificate authority as a feature. Androtracker mainly checks a serial number, checks suspicious behavior of SMS hiding, detects the malicious system commands in the code, and analyzes the suspicious permission requests.

(Muttoo, 2016) Applied a framework for identifying an abnormal Android application using permission and intent centered analysis. After fetching a number of permissions and intents as features from different downloaded applications, feature set was created and reduced using information gain algorithm. The feature set is then fractioned into two parts - the training data set and the testing data set. Then they used k-means clustering followed by the J48 and ID 3 classifier using the training data to properly distinguish malware that are present in the given applications. From the experiments conducted and data obtained thereof, they concluded that the J48 classifier produces the best results with an accuracy of 94 per cent. Performances were compared in terms of accuracy and error percentage derived from the confusion matrix.

Novel machine learning technique to detect malware by mining the patterns of permissions and API function calls acquired and used by android applications (**Mengyu Qiao, 2016**).Different features are combined, selected and experimented with machine learning methods, including Support Vector Machines, Random Forest and Neural Networks. Results show that their method obtains good accuracy in the finding of diverse types of Android malware, and show great abilities to soften malware threats for Android devices. The Android malware applications used are from the Android Malware Genome Project dataset consists of 1260.

Simple and low cost approach for detecting Android malware **(Ryo Sato, 2013**) uses simple features of manifest file. Manifest files are required in all Android applications, and thus, the proposed method is applicable to all Android applications. Experimental results they obtained shows that the offered technique can notice strange malware samples that are untraceable by a simple signature-based approach. Furthermore, the cost of exploring only the manifest file is really low. The dataset that they used, contain small number of samples; only 365 samples in total. The projected ay extracts six types of figures from manifest.XML files and uses them to spot Android malware.

Different approaches used in the research area in the dynamic analysis are described below:

Runtime Detection Framework for Android Malware (**TaeGuen Kim, 2018**) proposed a novel Android malware revealing structure that uses the application rewriting method for application behaviour checking without any kernel variations. Additionally, they considered the framework as a client-server model and set up the analysis processes for malware detection on the server-side. Client segment only keeps an eye of applications, and the server segment look at the traces by means of the suffix tree algorithm. In the detection process, they designed the suffix tree-based model to have the confidence values that are pre computed using the document frequency ratios and the likelihood values from the HMM. In addition, they also proposed the scoring method to make the final decision for the malware detection. They made efforts for our framework not only to have the high detection accuracy, but also to be efficient enough to examine the application behaviours at runtime.

Randrtoid **(J.D.Koli, 2018**) malware detection system is proposed which uses permission, APIs, and presence of others key apps information by using various machine learning techniques which can automatically distinguish malicious Android apps (malware) from legitimate ones. Randroid system employs various machine learning techniques i.e.; Support Vector Machine (SVM), Decision Tree (DT), Nave Bayes (NB) and Random Forest (RF) to perform malware classification. It makes use of comprehensive static analysis approach of application. The system uses permissions, API calls along with presence of key app's information. Experiment result shows that the proposed system is able to identify malware in accurate manner. It also verifies the fact that the use of mentioned information in feature set helps to achieve better result.

Malware Detection Methodology for Android systems using different features like System Call Logs. This paper (**Sanya Chaba Rahul Kumar Rohan Pant Mayank Dave, 2016)** defines an approach that creates a dataset using system call log information. The dataset is then implemented on three algorithms namely, Naive Bayes algorithm, Random Forest Algorithm and Stochastic Gradient Descent algorithm. The results are analyzed and accuracy is calculated. The main features of this algorithm include: Firstly, usage of system call logs i.e. working at the kernel level to find the malicious behavior of the applications. Secondly, the entire process is automated consuming a minimum amount of time. Thirdly, the quality of the dataset is improved by using a filtering algorithm called *Chi Square*. Lastly, the correctness and quality of the dataset is justified with the high accuracy results we obtained.

Identifying Android malware with system call co-occurrence matrices (**Xi Xiao, 2016**) mainly focus on the malware detection effect of system calls. From the experiments earlier, it can be seen the co-occurrence matrix built on the system call sequences extracts more useful information than the plain system call frequency vector. The co-occurrence matrix improves the TPR as well as decreases the FPR significantly. In this work, based on frequency vectors and co-occurrence matrices, they used KNN, logistic regression; naive Bayes, decision tree, random forest and SVM algorithms to classify Android malware. The malware dataset is offered by Zhou and Jiang, including 1227 malware samples that are divided into 49 malware families. By modifying the Chrome plugin Apk-downloader, they downloaded 1189 applications from the Google Play as the benign applications in experiments. At last, inspired by the boost of detection results with the co-occurrence matrix, they could improve the malware detection by extracting more meaningful regular patterns from the plain feature with information methods.

Classification preprocessing normalization of system call frequency, which improves the accuracy of Android malware detection, and by analyzing the results, **(Xiangli, 2016**) it can be summed that benign or malicious software, each type having its own discriminative system calls order list. In this paper, after extracting system calls from the app using strace command, they use normalization system call frequency for normalizing the system calls. It can be concluded that the number of one same system calls frequency of different applications varies hugely according to a list of collected system calls frequency, such as the frequency of general application calls ioctl function ranges between 10000-20000. However, some applications system calls frequency value may reach 200,000 times, or even more. This is also easily found in some other system calls. Because some system calls frequency is too large or too small, the system call frequency information is limited in accordance with the system call frequency specification sheet in order to reduce its effect.

III. MACHINE LEARNING CLASSFIER:

The most widely used machine learning classifiers for sentiment analysis training include the following classifiers:

A. Support Vector Machine

It is also called the Support Vector Networks. It is a non-probabilistic binary linear classifier but can be also used as a non-linear classifier using implicit mapping. It demonstrates linear-separability in high dimensions by using hyper planes. It performs the non-linear mapping of the input space into a high dimensional feature space this is also called the kernel trick.

When used for classification the SVM classifier constructs an optimal separating hyper plane in the high dimensional feature space. In this way it works like a maximal marginal classifier. When used for regression SVM classifier performs linear regression in the high dimensional vector space, further it does not penalize small errors (Introduction to Support Vector Machines.

.

SVM works to maximize the margin between the different classes. Hence, it plays the role of a discriminative classifier (**CORINNA CORTES, 1995**). The SVM is a supervised learning method that generates input-output mapping functions from a set of labelled training data. To compute the SVM classifier we need to minimize the SVM formula:

$$\left[1/n\sum_{i=1}^{n}\max\left(0,1-y_i(w.x_i-b)\right)\right]+\gamma||w||^2$$

B. Naïve Bayes

K-NN (**Blog, 2018**) is a relatively simple robust and versatile supervised learning algorithm that works in a non-parametric and instance based way. An advantage of using k-NN classifier is that it takes into account the local nature of the data. The formula for the most commonly used Euclidean distance parameter is calculated as follows:

$$D(x, x^{/}) = \sqrt{\{(x_1-x^{/}_1)^2 + (x_2+x^{/}_2)^2 + (x_3+x^{/}_3)^2 + \ldots + (x_n+x^{/}_n)^2\}}$$

In k-NN the training dataset is used to perform the classification of each of the members of the target dataset i.e. the set to be classified. For each of the entities in the target dataset, the k closest neighbours are selected and then the distance of the entity from each of these is measured mainly by using the Euclidean distance to ascertain how these neighbours will be affecting the classification of the entity in question.

IV. Android malware detection

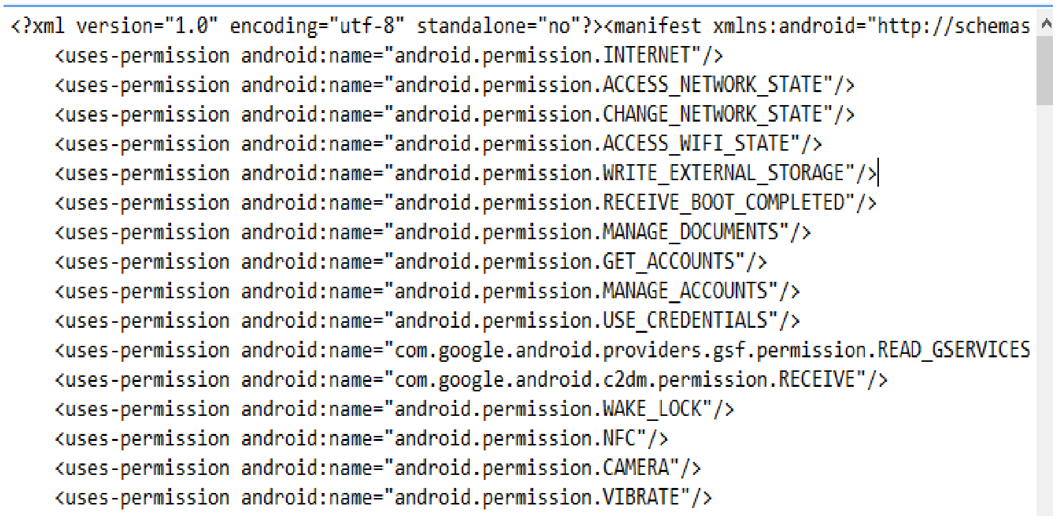
Android malware apps can be recognized alongside legitimate apps based on the permissions that the app requests during installation. There are two approaches of detecting an Android malware applications Firstly, the permission fields are extracted from the Android manifest file of the apps. Database of all the permissions for both normal and malware data is established and finally the machine learning algorithms are used to classify and identify the malware in Android applications. This comes in the approach of static malware detection, where we statically take out the features from an Android app. Second approach is, dynamic behaviours of an application are conducted by the system call sequences at the end. Hence, we observe the system call log of each application, use the same for the construction of our dataset, and finally use this dataset to categorize an unknown application as malicious or benign.

Two approaches of detecting an android malware

Android malware has developed rapidly in a tiny time and would keep rising at pace with our use of mobile devices. It becomes a very critical problem for us to make a distinction between Android malware from benign applications. There are two approaches of detecting an Android malware static and dynamic.

1. **Static Analysis Approach**

In this approach a predefined Android applications come in an Android package (.apk) archive. This .apkfile is nothing but a zip package of AndroidManifest.xml, classes.dex and other resources and folders. For extracting these features we originally need to reverse engineer the.apk files. This is done using the apktool (Virus Total, 2015). The AndroidManifest.xml file contains lot of features that can be used for static analysis. One of the foremost features used in almost all the research papers or antivirus scanners is the permissions requested by the application. The AndroidManifest.xml encompasses the number of permissions prerequisite by the application. In order to mine these permissions, systematic xml parsers cannot be used since Android has its own proprietary binary XML format. We designed a new xml parser competent of extracting permission feature from theAndroidManifest.xml file of the application.Fig1 shows the snapshot of Android manifest file.xml of YouTube app.



```
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://schemas
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    <uses-permission android:name="android.permission.MANAGE_DOCUMENTS"/>
    <uses-permission android:name="android.permission.GET_ACCOUNTS"/>
    <uses-permission android:name="android.permission.MANAGE_ACCOUNTS"/>
    <uses-permission android:name="android.permission.USE_CREDENTIALS"/>
    <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES
    <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-permission android:name="android.permission.NFC"/>
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.VIBRATE"/>
```

Fig1. Android Manifest file for youtube.apk

2. **Dynamic analysis approach**

In dynamic analysis, applications are conducted in a guarded environment to build a model that characterises the applications behaviours. In this approach run time behaviour of an Android application is retrieved by installing the app in the device and observes its behaviour. Therefore, the dynamic analysis methods cannot be simply evaded by trouble-free obfuscation or polymorphic techniques. A major amount of them has focused on modelling genuine program behaviours by system call frequencies. A system call is the mechanism through which a user interacts with the kernel in the operating system to demand an action to be performed. In order to accomplish this we have made use of the Android Emulator which comes along with the Android Studio (Install Android Studio, 2018). Here we execute each Android application in separate emulator called Genny motion and record the system calls as almost immediately as the application is installed in the emulator. This technique records the frequency of all the successful system calls recorded. The log file has proportion of the time consumed by the system call, seconds of the time for which the system call

v. Program of work

The overview of the objectives of the dissertation work experiments are listed in the table below:

Table 1: Objectives of Experiments

List of Experiments	Objectives
1. To design a dataset comprising of Static features using reverse engineering.	To use these features for identifying the malwares in the Android platform.
2. Installing apps in an environment for analysing android application.	To develop a method that will be used for detecting malwares
3. To train the various machine learning classifiers.	To use the classifiers SVM, NB.
4. To compare the performance of the machine learning algorithms	To analyse and evaluate the machine learning classifiers based on accuracy, recall rate and precision.

REFERENCES

Blog, K. Z. (2018). *A Complete Guide to K-Nearest-Neighbors with Applications in Python and R.* Retrieved from kevinzakka.github: https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

CORINNA CORTES, V. V. (1995). Support-Vector Networks. *Machine Learning, Kluwer Academic Publishers, Boston. Manufactured in The Netherlands*, 20,273-297.

Dave, S. C. (2016). Malware Detection Approach for Android systems Using System Call Logs . *springer*.

Handa, A. a. (2015). Malware detection using data mining techniques. *internationa;; journal of advanced research in computer and communication Engineering*, 5.

Hiranwal, K. M. (2013). A Survey on Techniques in Detection and Analysing Malware Executables. *International Journal of Advanced Research in Computer and Software Engineering ,vol.3no.4,*, 422-428.

How to use strace and itrace commands in Linux. (2018, 8 5). Retrieved from the geek dairy: https://www.thegeekdiary.com/how-to-use-strace-and-ltrace-commands-in-linux/

Hyun Jae Kang, J.-w. J. (2017). Androtracker:Creator Information based malware Detection. *International Conference on Technology on ioT*, 7.

Install Android Studio. (2018, 8 5). Retrieved from Developer Android: https://developer.Android.com/studio/install

J.D.Koli. (2018). RAndroid : An Android Malware Detectionusing random machuine Learning Classifiers. *IEEEE International Conference on Technologies for Smart -city Energy Security and power*.

Mengyu Qiao, A. H. (2016). Merging Permissions and API callsfor Android Malware detection. *5th IIAI International Congress on Advanced Informatics*.

Muttoo, S. V. (2016). An Android Malware Detection Framework based in Intents and Permissions. *defence Sceince Journal Vol 66,No.6*, 618-623.

RaviKiranVerma, K. P. (2017). Ansroid Malware detection and security using Machine learning. *international Conference on I_SMAC(IoT Social Mobile,Analytics and Cloud)*, 618-623.

Ryo Sato, D. C. (2013). Detecting Android Malware by Analyzing Manifest File. *Proceedings of the Asia-Pacific Advanced Network 2013 v. 36, p. 23-31. http://dx.doi.org/10.7125/APAN.36.4 ISSN 2227-3026*, 23-31.

Sanya Chaba Rahul Kumar Rohan Pant Mayank Dave. (2016). Malwarre Detection Approach for Android Systems using System Call logs.

TaeGuen Kim, B. K. (2018). Runtime detection Framework for Android Malware . *Hindawi Mobile Information Systems Volume 2018, Article ID 8094314, 15 pages https://doi.org/10.1155/2018/8094314*, 15.

Virus Total. (2015). Retrieved from virustotal.: https://www.virustotal.com/#/home/upload

Xi Xiao, X. X. (2016). identifying Android malware with system call co-occurrence matrices. *Transactions on emerging telecommunications technologies trans.Emerging tel.tech.2016,27*.

Xialoeiwang, Y. Z. (2015). Accurate malware detection in cloud . *springer plus*, 123.