Android mobile security by detecting and classification of malware based on permissions using machine learning algorithms

Ravi Kiran Varma P
Associate Professor
MVGR College of Engineering
ravikiranvarmap@gmail.com

Kotari Prudvi Raj
M.Tech, CNIS
MVGR College of Engineering
prudvirajkotari@gmail.com

K. V. Subba Raju
Assistant Professor
MVGR College of Engineering
srkakarlapudi@gmail.com

Abstract - **Android occupies a major share in the mobile application market. Android mobiles have become an easy target for the attackers. The main reason is the user ignorance in the process of installing and usage of the apps. Android malware can be detected based on the permissions it requests from the user. Several machine learning algorithms are being used in the detection of android malware based on the list of permissions enabled for each app. This paper makes an attempt to study the performance of some of the machine learning algorithms, viz., naïve Bayes, J48, Random Forest, Multi-class classifier and Multi-layer perceptron. Google play store 2015 and 2016 app data are used for normal apps and standard malware data sets are used in the evaluation. Multi-class classifier was found to be outperforming the other algorithms in terms of classification accuracy. Naïve Bayes classifier has outperformed as far as model construction time is concerned.**

Keywords: Android, permissions, Malware detection, classification, machine learning algorithms.

I. INTRODUCTION

In preset mobile market, android is the most famous platform for smartphones, the market growth rate is increasing gradually and it is now at 84.7% [1]. With the rapid development of mobile computing technology, tablets and smartphones have evolved sophisticated functions at lower costs. While, other platforms like iOS users allow to install applications only through iTunes, android allows many open sources, such as Torrents, Google play store, Direct downloads, or Third-party markets, etc. [1] This sovereignty makes distribution of applications and bundling of malware an easy task for attackers, [2] who try to lure the users into running malicious code. Malicious payload are used in repackaging popular apps [3]. 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. A lot of security measures have been implemented by the Android platform providers to stop malware during the installation, permission of Android system is most notable [4][22]. Android malware apps can be identified against genuine apps based on the permissions that the app requests during installation. This paper compares the performance of several machine learning techniques, viz., naïve Bayes, J48, Random Forest, Multi-class classifier and Multi-layer perceptron. The proposed method has mainly three stages. Firstly, the permission fields are extracted from the android manifest file of the apps. Second, a data base 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 [5], [6].

II. RELATED WORK

Schultz et al. [7] first proposed the concept of detecting malware in data mining, using navies Bayes machine learning algorithm. They have reported a result of 97.11%. One of the earlier works in android malware detection using application permissions are [8] [9] they have implemented machine learning algorithms for classification of android apps to detect malware. Firdausi et al. [10] Used J48 decision tree Machine learning algorithm for malware detection. They have reported a result of 97%**.** Sebastian et al. [11] using different attributes as intents. They have reported 96.02% with Random forest machine learning algorithm. Firewall and IDS also plays an important role in security. Varma et al. [12] Used Ant Colony Optimization (ACO) in detecting anomalies in firewall rule configuration. Varma et al. [13] used fuzzy-rough feature minimization and ACO search in optimization of feature selection for real-time Intrusion Detection System (IDS). Decision Trees are proved to be very effective in classification problems of network security [13]. Siddiqui et al. [14] used random forest machine learning algorithm for detecting malware. They have reported a result of 96.6%. Anderson et al [15] are used support vector machine (SVM) for malware detection. They have reported a result of 98.07%. There is a need to

identify the most performing technique in identification of malware in android app data. This study has compared several machine learning algorithms and identified the best algorithm.

III. DATASET

A total 3,258 android applications are collected, among them 2,999 are the normal apps and 1,999 are the malware apps. Google play store 2015 and 2016 app data are used for collecting normal apps [16] and standard malware data sets [17] are used in the evaluation.

A. Malware

Any software or a program with bad, unauthorized, and illegal behavior can be called as malware. A mobile user can hardly identify the intensions of the application based on its permissions. The users often install apps from unauthorized sources and invites trouble. The malware apps typically are programmed by the attackers to steal credentials of financial transactions, emails, social networking information, key stroke information, camera images, local file system information etc., [8] [18]. Therefore a mobile security application is needed that can detect and identify genuine and malignant apps based on permissions that the apps request.

B. Static Analysis

Static analysis is nothing but analyzing software before executing whether it is malicious are not. [19]In static analysis, the detection platforms include byte-sequence n-gram, string signature, control flow graph, syntactic library cell and operational code (opcode) frequency distribution etc. [9] this analysis as a major drawback of dynamic code loading and code obfuscation. [18] Before doing static analysis executable file has to be decrypted and unpacked.

C. Dynamic analysis

It is used to analyze the malicious code and application behavior. [8]This possible when it will be interacting with the system and this should be done in controlled environment. The major advantage of this analysis used to records the app behaviors and detects dynamic code loading during the run-time process. [19]Due to the overhead of executing the application, hard to implement comparatively static analysis. This paper deals with detection of android malware using the manifest file permissions that can help in dynamic detection of malware during installation.

IV. ANDROID APPLICATIONS AND ITS PERMISSION APPROACH

An android app is bundled in the form a n APK file. Every app when unpacked consists of a file called as [20] Androidmanifest.XML and this file also includes the list of permissions that it wants to access from the host mobile. Figure 1 shows an example of Androidmanifest.XML [21] file.

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera" />
<uses-feature android:name="android.hardware.camera.autofocus" />
<uses-permission android:name="com.android.vending.CHECK_LICENSE" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.MODIFY_PHONE_STATE" />
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.READ_LOGS" />
```

Figure 1 Extracting android emanifest.xml file

A. Feature extraction

In order to use machine learning algorithms in classification of android malware, first the permission data set of all the apps are to be extracted. Database is formed by placing a 1 if the permission is present in the Androidmanifest.XML file and a 0 if not [23]. The steps involved in extracting the permission data are:

Step1: Collect genuine android apps from google app store (2015) [23].
Step2: Collect malware app data from standard datasets [9].
Step3: Extract android manifest.XML file from the apps [8].
Step4: Collect the permission from each app and prepare excel sheet.
Step5: Convert the data into WEKA.ARFF format.

0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,normal

0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0
,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,1,0,0,0,1,0,1,0,0,
0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1
,0,0,0,1,0,0,0,0,malware

First of all, decompress the application package to extract the content. While doing this first three steps used to retrieve information from the source [4]. Here Androidmanifest.xml files are helpful for processing to extract data. A total of 3258 samples of android applications are used. The data sets are shown in Table 1.

Table 1 Sample datasets

S.no	Normal datasets	Malware datasets	Total number of datasets
1	700	300	1000
2	1000	500	1,500
3	1500	1000	2,500
4	1999	1259	3,258

V. *EXPERIMENTAL RESULTS*

Five machine learning algorithms, viz., Navies Bayes, J48, Random forest, Multi-class classifier, and Multi-layer perceptron are used for comparison and evaluation of the data sets. To evaluate the data efficiency construct a confusion matrix to compute the accuracy of given experiments. The followed metrics used by the confusion matrix:

True positive (TP): Number of good ware applications correctly identified.

False positive (FP): Number of malware applications correctly identified.

True negative (TN): Number of malware applications correctly identified.

False negative (FN): Number of good ware applications are wrongly identified.

True positive rate (TPR): Percentage of good ware applications correctly identified.

$$(TP/(TP+FN))$$

False positive rate (FPR): Percentage of malware applications wrongly identified.

$$(FP/(TN+FP))$$

Overall accuracy (ACC): Percentage of applications correctly identified.

$$\frac{TP+TN}{TP+TN+FP+FN}$$

Table 2 presents the experimental results of all the five classifiers for several data sets. From this table it can be observed that Multi-class classifier have performed better over other classification algorithms. The average accuracy was recorded as 99.81% that is the highest among other classifiers. However, Naïve Bayes classifier performed better as far as computational complexity is concerned. The Multilayer perceptron (MLP) neural network classifier performed pretty close to the multiclass classifier in android malware detection. However, the MLP classifier's computational complexity is very poor.

Tables 3 to 7 shows the confusion matrix obtained by various classifiers. Confusion matrix clearly depicts the performance of the classifiers. The multiclass classifier's confusion matrix is given in Table 6 and it can be seen that the False Positives are 0. There is only one False Negative. The True Positive and True Negative rates are 100%. Figure 2 shows the graphical representation of the classification accuracies of all the classifiers that are considered for comparison.

A comparison of similar works related to android malware detection are listed in Table 8. It can be observed that the multiclass classifier have outperformed in terms of classification accuracy when compared to other similar works in this domain.

Table 2 Performance evaluation of classifiers

S.no	Navies Bayes		J48		Random forest		Multi-class classifier		Multilayer perceptron	
	Acc.	sec.	Acc.	Sec.	Acc.	Sec.	Acc.	Sec.	Acc.	Sec.
Dataset-1	99.41%	0.13	98.82%	**0.01**	99.41%	0.06	**99.9%**	0.11	99.8%	140.04
Dataset-2	99.21%	**0.05**	98.43%	0.09	99.60%	1.4	**99.80%**	0.14	99.60%	267.05
Dataset-3	98.94%	**0.03**	98.82%	0.22	99.52%	1.56	**99.64%**	0.3	99.52%	449.32
Dataset-4	99.18%	0.05	99.09%	0.31	99.81%	2.79	**99.90%**	0.37	99.81%	570.8
Average	99.19%	**0.065**	98.79%	0.16	99.59%	1.45	**99.81%**	0.23	99.68%	356.8

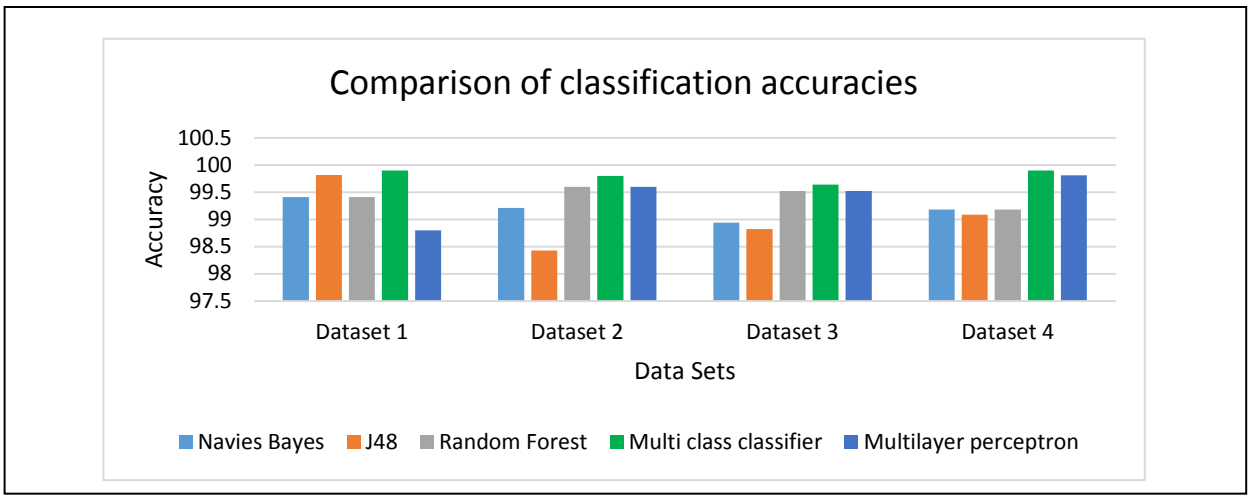

Figure 2 Graphical representation of performance evaluations

Table 3 Confusion matrix for Naïve Bayes

Navies Bayes	Normal	Malware
Normal	707	0
Malware	9	392

Table 3 presents the confusion matrix of the Naïve Bayes classifier. It indicates that 707 normal applications are classifier correctly as normal and none of them are classified as malware. That means there no false positives. However, there exist few false negatives.

Table 4 Confusion matrix for J48

J48	Normal	Malware
Normal	705	2
Malware	8	393

Table 4 is the confusion matrix resulted by J48 classifier. This classifier reported two false positives and eight false negatives. For this android malware detection problem domain, naïve Bayes classifier performed better than the J48 decision tree classifier.

Table 5 Confusion matrix for random forest

Random forest	Normal	Malware
Normal	707	0
Malware	2	399

Table 5 is the confusion matrix as resulted due to random forest classifier. This classifier reported zero false positives and two false negatives.

Table 6 Confusion matrix for multi class classifier

Multi class classifier	Normal	Malware
Normal	707	0
Malware	1	400

Table 6 lists the results of multi class classifier in the form of a confusion matrix. This classifier has outperformed all the other candidates under consideration for this work. The false positives are none and only one false negative is reported.

Table 7 Confusion matrix for multi-layer perceptron

Multi-layer perceptron	Normal	Malware
Normal	706	1
Malware	1	400

Table 7 shows the confusion matrix generated by MLP classifier. This has also performed comparatively better than the other ones. It has reported only one false positive and one false negative. Table 8 presents a comparative list of other similar works in this domain. Multiclass classifier has shown better performance compared to others.

Table 8 comparison with existed similar works

S.no	Author	Algorithm	Accuracy
1	Schultz *et al.* [7]	Navies Bayes	97.11%
2	Firdausi et al. [10]	J48	97%
3	Sebastian et al. [11]	Random forest	96.02%
4	Siddiqui et al [14]	Random forest	96.6%
5	Anderson et al. [15]	SVM	98.7%
6	This work	Multi class classifier	**99.9%**

VI. CONCLUSION AND FUTURE SCOPE

In this paper different machine learning algorithms are used such as navies Bayes, j48, Random forest, Multi class classifier and multilayer perceptron to detect android malware and evaluate the performance of each algorithm. Here we implement a framework for classifying android applications with the help of machine learning techniques to check whether it is malware or normal application. To access this model have to extract several features and permission from many downloaded applications from the android market. For validating, our system 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. From the experiment, it was found that multi class classifier has performed well over other methods, regarding classification accuracy. As far as computational complexity is considered Naïve 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 [24].

Feature reduction for android malware app data sets and performance evaluation shall be considered in the future work.

REFERENCES

[1] Madihah Mohd Saudi and Zul Hilmi Abdullah, "An Efficient Framework to Build Up Malware Dataset," *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 7, pp. 1104-1109, 0ct 7 2013.

[2] Abhay pratap singh and S.S handa, "Malware detection using data mining techniques," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 5, pp. 375-380, May 2015.

[3] Xialoei wang, yuexiiang, and zeng yinghi, "Accurate malware detection and classification in the cloud," *Spinger plus*, pp. 1-23, 2015.

[4] Kirti Mathur and Saroj Hiranwal, "A Survey on Techniques in Detection and Analyzing Malware Executables," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 4, pp. 422-428, April 2013.

[5] S. P. Choudhary and Deepti Vidyarthi, "A Simple Method for Detection of Metamorphic Malware using," *ScienceDirect*, pp. 265-270, 2015.

[6] Babu Rajesh V, Phaninder Reddy, Himanshu P, and Mahesh U Patil, "Androinspector: A System for comprehensive analysis of android applications," *International Journal of Network Security & Its Applications (IJNSA)*, vol. 7, pp. 1-21, September 2015.

[7] M., Schultz, E., Eskin, F., Zadok, and S. Stolfo, "Data Mining Methods for Detection of New Malicious Executables," *Proceedings of 2001 IEEE Symposium on Security and Privacy*, pp. 38-49, may 2001.

[8] Mayuri Magdum and Sharmila.K.Wagh, "Permission Based Android Malware Detection System using Machine Learning Approach," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 14, no. 2016.

[9] Zarni aung and zaw win, "Permission-based Android malware detection," *International journal of scientific & technology & research*, no. 3, march 2013.

[10] I., Firdausi, C., Lim, and A. Erwin, "Analysis of Machine Learning Techniques Used in Behavior Based MalwareDetection," *Proceedings of 2nd International Conference on Advances in Computing, Control and Telecommunication Technologies (ACT)*, pp. 201-203, December 2010.

[11] Sebastian Hahn, Mykolai Protsenko, and Tilo Muller, "Comparative Evaluation of Machine Learning-based Malware Detection on Android," *Lecture Notes in Informatics (LNI)*, pp. 79-88, 2016.

[12] Ravi Kiran Varma P, Valli Kumari V, and Srinivas Kumar S, "Ant Colony Optimization-based Firewall Anomaly Mitigation Engine," *Springerplus*, vol. 5, no. 1, pp. 1-32, 2016.

[13] Ravi Kiran Varma P, Valli Kumari V, and Srinivas Kumar S, "Feature selection using relative fuzzy entropy and ant colony optimization applied to real-time intrusion detection system," *Elsevier Procedia Computer Science*, vol. 85, no. 2016, pp. 503-510, 2016.

[14] M., Siddiqui, M.C., Wang, and J. Lee, "Detecting Internet Worms Using Data Mining Techniques," *Journal of Systemics, Cybernetics and Informatics*, pp. 48-53.

[15] B., Anderson, C., Storlie, and T Lane, "Improving Malware Classification: Bridging the Static/Dynamic Gap.," *Proceedings of 5th ACM Workshop on Security and Artificial Intelligence (AISec)*, pp. 3-14, 2012.

[16] APK Downloader. [Online]. https://apps.evozi.com/apk-downloader/

[17] Ridhima Seth and Rishabh Kaushal, "Dissecting Android Malware: Characterization and Evolution," *IEEE*, 2012.

[18] Imtithal A. Saeed, Ali Selamat, and Ali M. A. Abuagoub, "A

Survey on Malware and Malware Detection Systems," *International Journal of Computer Applications*, vol. 67, April 2013.

[19] Prajakta D. Sawle and Prof. A. B. Gadicha, "Analysis of Malware Detection Techniques in Android," *International Journal of Computer Science and Mobile Computing*, vol. 3, pp. 176-182, March 2014.

[20] Tarang Kumar Barsiya, Dr. Manasi Gyanchandani, and Dr. Rajesh Wadhwani, "Android malware analysis: A survey," *International Journal of Control, Automation, Communication and Systems (IJCACS)*, vol. 1, July 2015.

[21] Mohsen Damshenas, Ali Dehghantanha, and Ramlan Mahmoud, "A survey on malware propagation, analysis and detection," *International Journal of Cyber-Security and Digital Forensics (IJCSDF)*, pp. 10-29, 2013.

[22] Khalid Alfalqi, Rubayyi Alghamdi, and Mofareh Waqdan, "Android Platform Malware Analysis," *(IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 6, pp. 140-146, 2015.

[23] Saba Arshad, Abid Khan, Munam Ali Shah, and Mansoor Ahmed, "Android malware detection and protection: A survey", *(IJACSA) International Journal of Advanced Computer Science and Applications,* vol. 7, pp. 463-475, 2016.

[24] Ravi Kiran Varma P, Valli Kumari V, and Srinivas Kumar S, "A novel rough set attribute reduction based on Ant Colony Optimisation," *International Journal of Intelligent Systems Technologies and Applicaitons*, vol. 14, no. 3/4, pp. 330-353, 2015.