Reduced Permissions Schema for Malware Detection in Android Smartphones

Ahmed H. Mostafa, Marwa M. A. Elfattah and Aliaa A. A. Youssif

***Abstract*—** Day after day the dependence on smart devices is increasing, especially smart phones. As, smartphone is not just a phone device but also it is smart TV, GPS, smart camera and tablets, with expansion in the use of mobile in critical tasks such as online banking services, business transactions, and storing critical information such as credit cards, passwords and personal data, the malware's attacks are increased. Most of current malware detection solutions for mobile devices can detect known malware but cannot detect newfangled malware and others malware detection techniques depend on monitoring the behavior of the malware but the monitoring on the Smartphone can be a very heavy consuming task.

Hence, there is a need to develop a mobile malware detection that can provide an effective solution to protect the mobile user from any malware and at the same time address the limitation of mobile devices environment. In this paper we focused on extracted android system permissions from android applications .apk files. The research focused in reducing the number of android permissions to be used as features for machine learning classifier to detect android malware application.

***Keywords*—** Android, Smartphones, Malware Detection, Machine learning.

I. Introduction

Now a day's, smartphone is very popular and widely used in business and personal life. A lot of mobile phone users are rapidly switching to smartphones. According to eMarketer [1], it is expected that around 49% of the mobile phone users globally are likely to use smartphones by 2017. The great popularity of the smartphones is because of their powerful capabilities such as video calling, capturing images, recording video, playing digital media, sending and receiving emails, web browsing and access online banking services capturing images, recording audio and video, video calling, playing digital media, sending and receiving emails, web browsing, using social networks such as Facebook and Twitter, and communicating using Bluetooth and WIFI.

Smartphone users save important data on their phones, such as phone numbers, SMS messages, photos, passwords, credit card numbers, therefore smart phones are a very interesting target for attackers and malicious software. One important characteristic of smartphones that its ability to install third-party applications from many markets whether official or non-official. Unfortunately, there is no control on the non-official markets, therefore attackers can upload their application whether games, media or others applications to these markets and attempt to embed malicious program into benign applications

Many users download mobile applications without any thought of security. Whereas, with the rapid increase in the use of smartphones, the number of mobile applications is increasing, and according to PortioResearch [2] downloading of mobile applications will continue to grow to exceed 200 billion applications by the end of year 2017, The number of markets which allow users to download applications are increasing and the number of non-official market are also increasing but non-official market do n ot impose security measures on the phone applications that are being uploaded by developers so many hackers upload malicious applications to these markets

Actually, most smartphones users download mobile applications without any attention of security issue. Therefore, it is important to use a methodology to detect the malware applications before installing it on the phone.

The problem of detecting malware for smartphone presents a lot of challenges due to limited resources availability. Smartphones have limited hardware capabilities in comparison to the hardware capabilities of traditional computers, as smartphones have limited memory, and limited battery energy. So, current solutions for computers may not be applicable on smartphones.

Moreover, most current malware detection techniques depend on extract signatures pattern for malwares, and all malwares signatures are stored in repository. This repository represents malwares signature database to identify malware. The antivirus should search in the database for matching signatures, but it cannot detect new malwares.

On other hand, the other set of techniques depend on monitoring the behavior of malware during the run time but monitoring can be a very heavy consuming task [3]-[5].

Monitoring can be performed on remote servers but it is dependent on external server, which means there can be server down problems and network congestion [6].

With very rapid development in smartphones also its operating system have evolved so the techniques and approaches that applied on the previous Mobile operating

This work was supported by Computer Science Department, Faculty of computers and information, Helwan University.

Ahmed Hesham Mostafa, Computer science Department, Faculty of computer and information, Helwan university, Cairo, Egypt (phone: +2001095906541; e-mail: ahmedheshamostafa@gmail.com).

Dr. Marwa Mohamed Abd El Fatah, Computer science Department, Faculity of computer and information, Helwan university, Cairo, Egypt (e-mail: marwa_8_80@yahoo.com).

Prof. Aliaa Youssif , Computer science Department, F aculty o f computer and information ,Helwan University ,Cairo , Egypt (Phone: +202 27644827 ; Fax : +202 25547975 ;e-mail: aliaay@yahoo.com)

system need to be modified to be applicable with current operating system.

The most common mobile operating systems are Android, Blackberry, iOS, Windows Phone and Symbian.

Statista [7] expected that Android is expected to account for 62.4 percent of global tablet shipments in 2017, thus taking over as the market leader. Statista also expected that the smartphones deploying Android as operating system are forecast to reach around 1.5 billion units by 2018[8]. Cisco security report for 2014 finds 99% of all new mobile malware is targeting Android [9]. Android's Google Play store has officially reached over 1 million applications, and applications download have also grown to over 50 bi llion [10]. Several third-party Android Marketplaces exist without restricted security rules for submit applications.

The challenge of how to detect smartphones malwares depends mainly on how to extract the application features. Those features are then used to categorize the application as malware or as benign application. This research introduces a mechanism to select reduced number of application features, which are used in anomaly detection system to detect android malwares before installing it on smartphone.

The rest of the paper organized as follows. We start in section II with a brief background on malware detection techniques, in section III a survey of previous relevant studies. , in section IV brief background on android operating system , in section V describes the methods we used to collect data , extract features and building the dataset , in sections VI , VII we present the experiments and the evaluation results. Finally in section VIII discuss the results and conclusion.

II. Malware Detection techniques

There are two main categories of smartphones malware detection techniques, which are static detection techniques and dynamic detection techniques [3]-[5]. The major difference between static and dynamic analysis is how the data is acquired.

Static detection represents an approach of checking source code or compiled code of applications before it gets executed. It identifies malicious code by unpacking and disassembling the application to extract features for anomaly detection. It can use simple pattern search operation or slightly more complex machine learning approaches in order to detect weakness in the code of software.

On other hand, dynamic set of techniques identify malicious behaviors after executing the application on an emulator or controlled environment.

Static based techniques are fast, flexible and easy to be automated, which means, they are suitable for mobile devices whereas, in dynamic based analysis the monitoring can be a very heavy consuming task. Also, in dynamic based analysis, the malware can change his behavior during rum time and cannot be detected.

On other hand, there is different identification techniques depending on the type of identification carried out, detection systems can be classified as either anomaly-based, signature based system.

Anomaly-based identification attempts to model normal and non-normal behaviors during the training phase. Anomaly detection techniques have the potential to detect newfangled malware. However, they are prone to detect rare legitimate behaviors as malicious [5].

Signature-based identification aims at identifying known malicious by means of predefined patterns of signatures. The main benefit of signature detection lies in its accuracy detecting well-known attacks. In this regard, maintaining an up-to-date database with a massive amount of signatures poses a major challenge. Furthermore, resource-constrained devices are not capable of processing big amount of signatures [5]. Also, they need human expertise to develop new malware signatures, which is time consuming.

Static signature-based technique is very efficient and reliable to identify known malwares; otherwise, they cannot detect unknown malwares. Signatures must be up-to-date that lead to a massive amount of signatures. On other hand, anomaly based techniques have the ability to detect unknown malwares.

III. Related work

Crowdroid [11] and MADAM [12] are among the research works that perform android malware detection by monitoring the dynamic malware behavior through the system call. The drawback of this method is the high energy consumption as monitoring system calls consume lots of resources of a mobile device. Yerima [13] proposed approach based on Bayesian classification models obtained from static code analysis to detect android malware. Borja Sanz [14] proposed PUMA which they extract permission to train machine learning algorithm and they use all permissions and the best accuracy result with RandomForest is 0.8637. Xing Liu proposed a two-layered permission based detection scheme for detecting malicious Android applications [15].

IV. Android system architecture

Android [16] is open source OS built on Linux for mobile devices. As shown in Fig. 1, Android system consists of:

- Linux kernel provides basic system services, such as process scheduling.
- Intermediate layer include Android native libraries and Android runtime environment.
- Android native libraries include core libraries such as the system C library, media libraries; various system components in the upper layers use these libraries.
- Android runtime environment is the Dalvik virtual machine.
- Application Framework layer is which make it easy for developers to develop new applications.
- Application layer include core applications, such as call, message and third-party developed applications.

ISBN: 978-1-61804-320-7

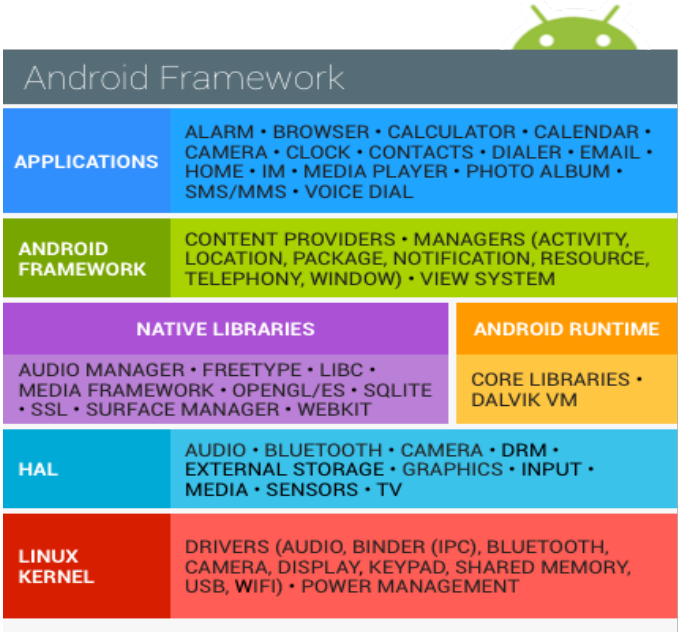

Fig. 1 Android Software Stack [17]

Android applications are developed with Google Android SDK [18] and written in Java language. Then the source code is compiled into .dex file, and packaged in an .apk archive for installation.

Android permits application installation from third party vendors mean that Google has no control over the quality or safety of the applications provided in these stores.

Android Permissions is a critical design point of the Android security architecture is that no application has permission to perform any operation that would impact other applications, the operating system, or the user , this includes reading or writing the user's private data (such as contacts or e-mails), reading or writing another application's files, performing network access etc [19].

Android sandboxes applications from each other so, application must explicitly share resources and data. They do this by declaring the permissions they need for additional capabilities not provided by the basic sandbox. Applications statically declare the permissions they require, and the Android system prompts the user for consent at the time the application is installed [19].

V. Research Methods

The analysis of applications is often to classify an application as malicious or benign. In classification features are used to make decisions. Application features are required to be informative to produce an accurate decision.

In many real-world applications, numerous features are used in an attempt to ensure accurate classification. If all those features are used to build up classifiers, then they operate in high dimensions, and the learning process becomes computationally and analytically complicated. Hence, there is a need to reduce the dimensionality of the feature space before classification.

This work mainly aims to extract android application features based on dimensionality reduction technique that extracts a s ubset of new features from the original set of features by means of some functional mapping keeping as much information in the data as possible.

A. Collect Data

To conduct experiments, a dataset of real Android applications and real malware in considered. In particular, an initial dataset of 325 malware and 325 be nign android applications is acquired. The malwares are collected from Contagio Malware Dump [20] Android Malware Dump [21] and MalShare [22].

Malware applications represent more than 89 android malware families [23], [24]. Whereas, the benign applications cover all android categories in Google play store [25].

B. Extract Features

Android permissions control the access to sensitive resources and functionalities. Permissions allow an application to access potentially dangerous API functionality. Many applications require several permissions to function properly. These permissions must be listed explicitly in the application's Manifest.xml file. Every application must have an android Manifest.xml in its root directory. The manifest presents essential information about the application to the Android system.

Using the permissions as features for machine learning classifier can help to detect the malware before the installation. So, analyzing the android applications manifest files to identify the permission set requested by that application can considered as an informative methodology for anomaly based feature extraction in static manner.

First of all, the application .apk file is decompressed to retrieve the content. All permissions used by each APK file are extracted statically using python [26] script that is developed based on AndroGuard API [27]. We developed a python script to automate the extraction of the features. The developed script unpack the apk files to classes.dex and the manifest file in binary format, then convert the manifest to xml file, where, all permissions used by the application can be extracted.

All permissions from manifest files are extracted based on the following methodology:

- Vector V contains all android system permissions.
- For each application there is features vector V_i contains all features for each application the feature vector represents all android system permissions. So, for each application a_i in the Applications set A there is binary vector $V_i = \{v_1, v_2, v_3, ..., v_n\}$ where, n is number of permissions available in the Android system, and,

$$v_i = \begin{cases} 1, & \textit{if exxtracted permission } v_n \textit{ exist in } V \\ 0, & \textit{else} \end{cases}$$

- The variable C is the type of the applications to be benign or malware where $C \in \{Malware, Benign\}$
- The creating of matrix M process is described by following algorithm :

Input: *set A contain all apk files and vector V contain all android system permissions*
Output: *matrix M contain all vectors* V_i

for each a_i *in A* ***do***
  Extract all permissions from a_i *and set it to set* S_i
  for each $s_j \in S_i$ ***do***
    if $s_j \in V$ ***do***
      $v_n \in V_i = 1$
    else
      $v_n \in V_i = 0$
    end if
  end for
  Set V_i *in M*
end for

After applying the previous methodology on all of the collected dataset, we noted, that the benign application use 1141 permissions and malware application use 4882 permissions. Approximately, malware applications use nearly three times permissions more than benign applications, that means malwares actually use permissions to access functions the benign applications not use.

C. Reducing number of features

Actually, we count 151 android system permissions according to android 5.0 Lollipop with API level-21 [28],[29] considering all of android permissions as a feature set will produce an enormous feature vector for each application. So it is required to reduce the number of the application features, where the high dimension data makes testing and training of general classification methods complicated.

The goal of data reduction is to find a minimum set of features such that the resulting probability distribution of the data classes is as close as possible to the original distribution obtained using all features. Using the reduced set of features has additional benefits. It reduces the number of features appearing in the discovered patterns, helping to make the patterns easier to be understood. Further it enhances the classification accuracy and learning runtime.

In the conducted experiments, applying the previously stated methodology for feature extraction based on android permissions produced a matrix M, which contains the vectors of the android system permissions of all collected applications. For reducing the feature set, a preprocessing step has been performed, which is removing all zero-frequency-permissions in the binary matrix M . The permissions that its frequency is zero are those which are not used by any malware or benign applications, the number of features were reduced to 114 features.

Then, to select the most informative feature set, two feature selections methodologies are applied [30],[31], namely information gain and Gain Ratio based feature selection methods. The information gain and Gain Ratio score are calculated for each permission - attribute in M matrix- that is for telling how important a given attribute of the feature vectors is.
InfoGain and GainRatio are calculated as following:

$$InfoGain\ (C, v_n) = H(C) - H(C \mid v_n) \quad (1)$$

$$GainRatio\ (C, v_n) = (H(C) - H(C \mid v_n)) / H(v_n) \quad (2)$$

Where H is the information entropy, Y and X are random variables and P is the probability.

$$H(X) = \sum_i P(x_i) I(x_i) = -\sum_i P(x_i) \log_b P(x_i) \quad (3)$$

Where the conditional entropy of two events *X* and *Y*

$$H(X \mid Y) = \sum_{i,j} P(x_i, y_j) \log \frac{P(y_j)}{P(x_i, y_j)} \quad (4)$$

After calculating the score for each permission, all permissions that its score is zero are removed. Two sets of features (permissions) have been produced, the first set of permissions calculated by InfoGain shown in Fig. 2, and the second set calculated by GainRatio shown Fig. 3.

D. The best reduced dataset

Now there are three datasets:

- First, Dataset#1, which is based on features permissions ranked by InfoGain, as shown in Fig. 2.
- Second, Dataset#2, which is based on features permissions ranked by GainRatio, as shown in Fig. 3.
- Third, the original Dataset based on all permissions.

To select the best reduced features set, WEKA [32] tool is used to evaluate the two sets against different classifiers.

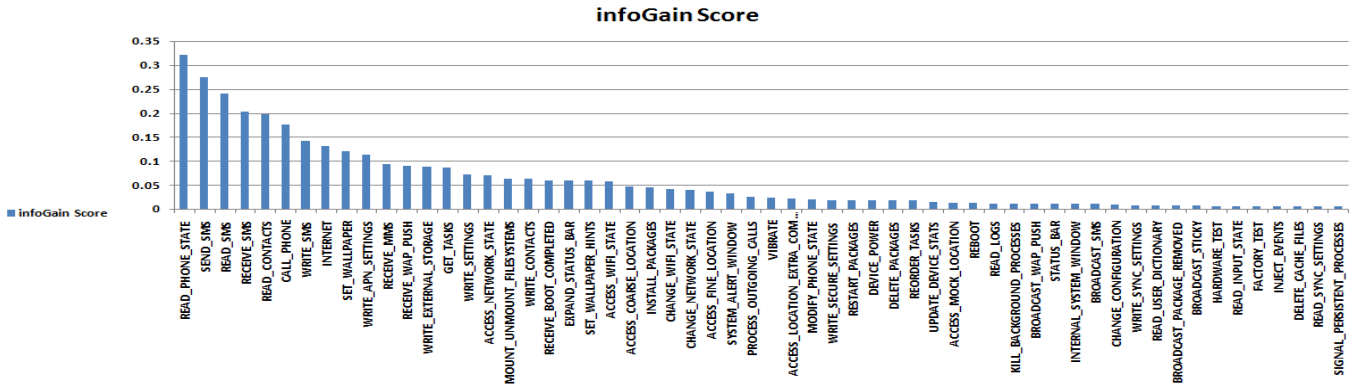

Fig. 2 Top 58 ranked feature using InfoGain

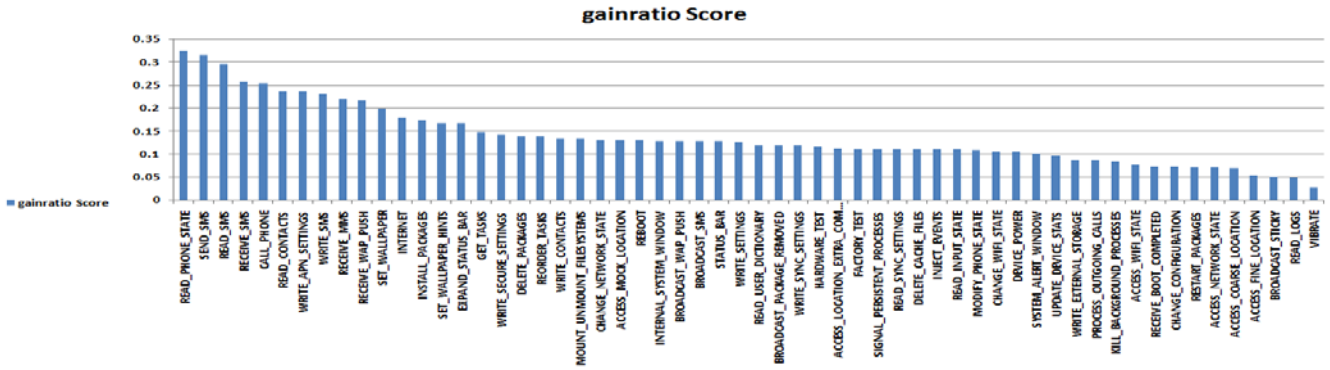

Fig. 3 Top 58 ranked feature using GainRatio

VI. Experiments

WEKA tool is used to evaluate the three data sets against different classifiers ,the results were compared that was obtained f rom all experiments form dataset based on all permission and Dataset#1, which is based on f eatures permissions ranked by InfoGain Dataset#2, which is based on features permissions ranked by GainRatio

A. Classifiers

More than one classifiers from WEKA are used to evaluate the best features set. The used classifiers are: C4.5 algorithm (J48), feed forward artificial neural network (MultilayerPerceptron MLP), Support vector machine (LibSVM) , Radial Basis Function Network(RBFClassifier) , stochastic gradient descent (SGD) , Logistic Regression (Logistic) , ExtraTree , J48Consolidated , RandomForest Tree , RandomTree , K- nearest neighbours classifier (IBk) , KStar and best-first decision tree (BFTree)

B. Testing Options

WEKA has different mechanisms to divide the experimental dataset into training dataset and testing dataset testing that is used to train and test the classifiers models. The first methodology is k-cross validation [33]. In k-fold cross-validation, the dataset is randomly partitioned into k equal size subsamples. One subsample is used as the validation data for testing the model, and the remaining k-1 subsamples are used as training data. Then repeated k times, with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged to produce a single estimation. In the conducted experiments, two k values have been chosen, k=10 and k=3 folds.

Another mechanism is simply to divide dataset into two portions in a random manner. Here, 66% of the original dataset are randomly chosen for training, and the remaining 34% of the data are used for testing and the last testing option is the use of training data as the testing data.

C. Measure of classifiers

The evaluation was performed by measuring the following metric:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (5)$$

The *Accuracy* is the percentage of predictions that is correct , where TN is the number of benign applications correctly classified, TP is the number of malware cases

ISBN: 978-1-61804-320-7

correctly classified, FP is the number of benign applications incorrectly detected as malware, and FN is the number of malware incorrectly classified as benign applications (false negatives).

D. *Experiments steps*

First, the classifiers was trained using Dataset#1 that is based on reduced features selected by InfoGain, see Fig. 4.

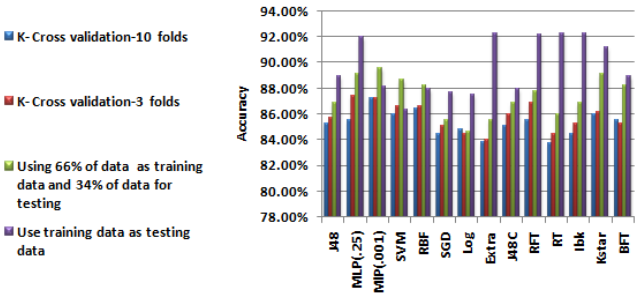

Fig. 4 the Accuracy for classifiers trained by Dataset#1 selected by InfoGain

Then, the classifiers was trained using Dataset#2 that is based on reduced features selected by gain ratio, see Fig. 5.

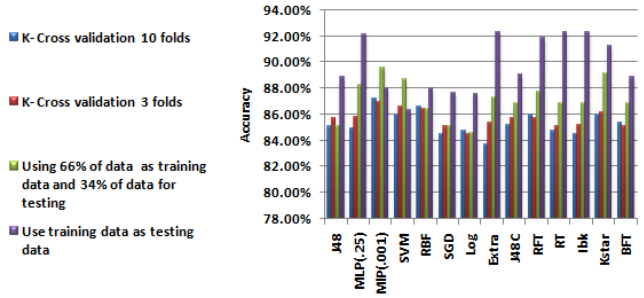

Fig. 5 the Accuracy for classifiers trained by Dataset#2 selected by GainRatio

Finally, the classifier was trained using original Dataset with all permission, see Fig. 6.

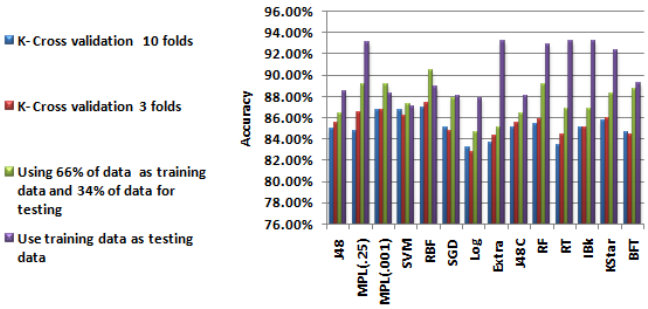

Fig. 6 The Accuracy for classifiers trained by Dataset with all permissions

VII. RESULTS

In this section, some of results obtained from previous experiments are concluded.

Fig. 7 show the results of the experiments conducted using k = 10 f old. The Fig. 7 show that in most of cases the classifications using the two reduced datasets give results better than that the classification using the dataset#3 gives. For example, with classifier MPL with learning rate .001 the reduced datasets give better result than using all permission, the classifier MPL (.001) with infoGain and RatioGain give accuracy 87.2308 % and with all permission give 86.7692 % and the best result given by all permissions given with RBF classifier 86.9231 %

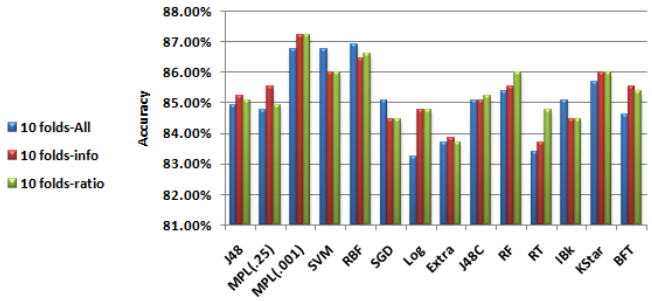

Fig. 7 Accuracy of classifiers using all permission and reduced permissions by InfoGain and GainRatio 10 folds

Fig. 8 show the results of the experiments conducted using k = 3 fold. Also, It is widely noted that classifications using the reduced datasets give results better that that given by classification using dataset#3 in most cases expect with SVM and RBF whereas the best result given with infoGain is the result given by MPL with leaning rate 0.25 is 87.3846 %, the best result given by GainRatio is given by MPL with learning rate .001 is 86.9231 % and the best rest result given by using all permission is given by RBF is 87.3846 %.

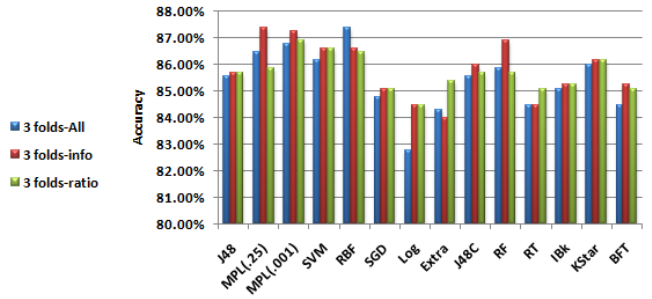

Fig. 8 Accuracy of classifiers using all permission and reduced permissions by info and GainRatio

But with using 34% as the testing option it is noted that in some classifier using all permissions give better result than reduced permissions and with other classifier the reduced permission give similar or better result than all permission see Fig. 9 whereas the best result given with infoGain and Gainratio is the result given by MPL with leaning rate 0.001 is

89.5928 %, and the best rest result given by using all permission is given by RBF is 90.4977 %.

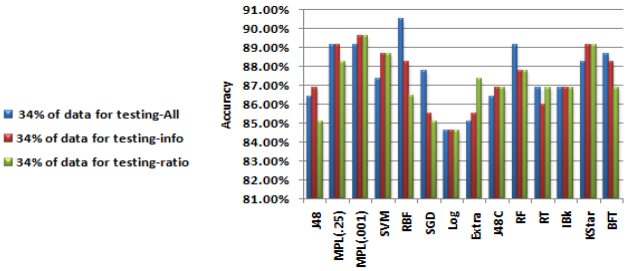

Fig. 9 Accuracy of classifiers using all permission and reduced permissions by info and ratio gain

So we can note from previous results in most cases especially with testing options 10 and 3 folds when we use the reduced permissions give equivalent or better results than using all permissions and with 34% as testing option the results are in with most classifiers are similar, so the reduced features set obtained by InfoGain or GainRatio can be used instead of using all android permissions as features for distinguish between benign and malware application.

The comparison between results obtained by info gain and gain ratio show that the accuracy for classifier trained by the feature set selected by info gain and gainratio are very similar as shown in Fig. 10 .

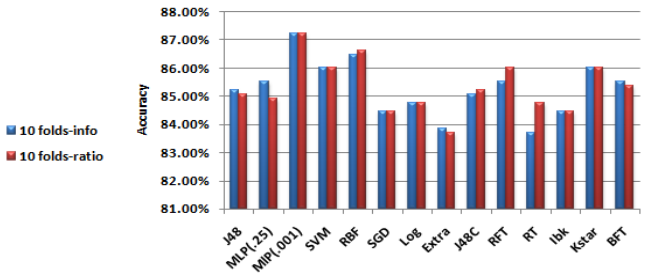

Fig. 10 InfoGain results against GainRatio result using 10 folds

But with the 3 fold and 34% testing options the result show the InfoGain result is better than the ratio results Except GainRatio give better results than InfoGain with RT and Extra Trees classifiers as shown in Fig. 11 and Fig. 12.

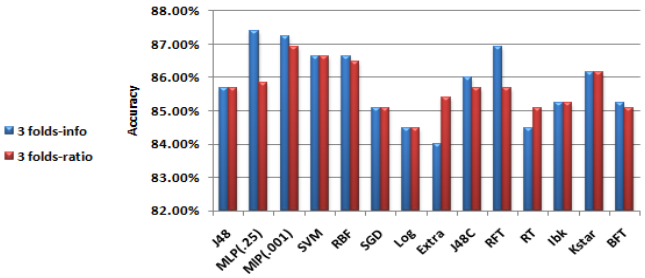

Fig. 11 InfoGain results against GainRatio result using 3 folds

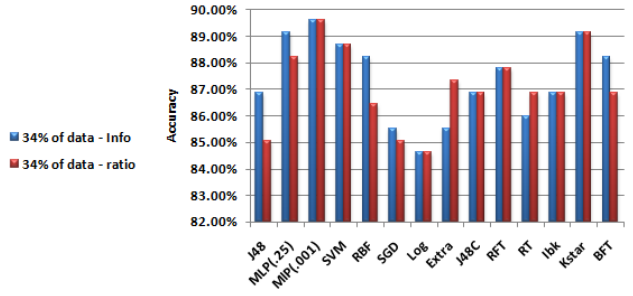

Fig. 12 InfoGain results against GainRatio result using 34% of dataset as a testing set

VIII. Conclusion

In this paper we used android system permissions as features that extracted from Android application (.apk) files. The extracted data is used as features during a classification process of the applications. We focused on reducing number of features by selecting the best permissions that can distinguish between malware and benign applications.

We collected 325 android malware application and 325 benign applications and extracted the permissions and we reduced the number of permissions to 58 instead of using 151 permissions and select permissions based on InfoGain and GainRatio and test the two different set against different classifiers.

We concluded that the reduced permissions obtained by InfoGain or GainRatio , that extracted statically from .apk files, coupled with Machine Learning classifier can provide good indication about the nature of an .apk file without running it on the smartphone and it can be used instead of using all android permissions as features for machine learning classifier to distinguish between benign and malware application where the best result given by reduced permissions whether InfoGain or GainRatio is 87.2308 % and the best result obtained by using all permissions is 86.9231% .In the future work we will extract more features from applications to be combined with the reduced permissions .

Acknowledgment

We would to thanks everyone help us to complete this research and i would to thanks my supervisors Prof. Aliaa and Dr. Marwa for their excellent guidance.

References

[1] Smartphone Users Worldwide Will Total 1.75 Billion in 2014 [Online]. Available: http://www.emarketer.com/Article/Smartphone-Users-Worldwide-Will-Total-175-Billion-2014/1010536

[2] Mobile Application Futures 2013-2017 [Online]. Available: http://www.portioresearch.com/en/mobile-industry-reports/mobile-industry-research-reports/mobile-applications-futures-2013-2017.aspx

[3] Abdelfattah Amamra, Chamseddine Talhi, and Jean-Marc Robert," Smartphone malware detection: From a survey towards taxonomy". In Malicious and Unwanted Software (MALWARE), 2012 7th International Conference on, pages 79–86. IEEE, 2012.

[4] Mariantonietta La Polla, Fabio Martinelli, and Daniele Sgandurra," A survey on security for mobile devices" Communications Surveys & Tutorials, IEEE, 15(1):446–471, 2013.
[5] Suarez-Tangil, Guillermo, et al. "Evolution, detection and analysis of malware for smart devices." Communications Surveys & Tutorials, IEEE 16.2 (2014): 961-987
[6] Yan Ma and Mehrdad Sepehri Sharbaf. "Investigation of static and dynamic android anti-virus strategies". In Information Technology: New Generations (ITNG), 2013 Tenth International Conference on, pages .403–398IEEE, 2013
[7] Worldwide market share forecast of smartphone operating system from 2010 to 2015 [Online]. Available: http://www.statista.com/statistics/266970/market-share-forecast-of-smartphone-operating-systems-from-2010-to-2015/
[8] Global Smartphone unit shipments forecast by operating system 2014 and 2018 [Online]. Available : http://www.statista.com/statistics/309448/global-smartphone-shipments-forecast-operating-system/
[9] Cisco: 2014 Cisco Annual Security Report [Online]. Available: http://www.cisco.com/web/offers/lp/2014-annual-security-report/index.html
[10] Android's Google Play beats App Store with over 1 million apps, now officially largest[Online]. Available: http://www.phonearena.com/news/Androids-Google-Play-beats-App-Store-with-over-1-million-apps-now-officially-largest_id45680.
[11] Burguera, Iker, Urko Zurutuza, and Simin Nadjm Tehrani. "Crowdroid: behavior-based malware detection system for android." Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices. ACM, 2011.
[12] Dini, Gianluca, et al. "Madam: a multi-level anomaly detector for android malware." Computer Network Security. Springer Berlin Heidelberg, 2012. 240-253.
[13] Yerima, Suleiman Y., Sakir Sezer, Gavin McWilliams, Igor Muttik. "A new android malware detection approach using bayesian classification." *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on*. IEEE, 2013.
[14] Sanz, Borja, et al. "Puma: Permission usage to detect malware in android." International Joint Conference CISIS'12-ICEUTE´ 12-SOCO´ 12 Special Sessions. Springer Berlin Heidelberg, 2013.
[15] Liu, Xing, and Jiqiang Liu. "A Two-Layered Permission-Based Android Malware Detection Scheme." Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2014 2nd IEEE International Conference on. IEEE, 2014.
[16] Android Operating System [Online]. Available : https://www.android.com/
[17] Android Security [Online]. Available : https://source.android.com/devices/tech/security/
[18] Android SDK [Online]. Available : http://developer.android.com/sdk/index.html
[19] Android system Permissions [Online]. Available : http://developer.android.com/guide/topics/security/permissions.html
[20] Contagio Mobile Mini Malware Dumb [Online]. Available : http://contagiominidump.blogspot.com/
[21] Android Malware Dump [Online]. Available: http://androidmalwaredump.blogspot.com/
[22] MalShare [Online]. Available : http://malshare.com/
[23] Cooper, Vanessa N., Hossain Shahriar, and Hisham M. Haddad. "A Survey of Android Malware Characterisitics and Mitigation Techniques." Information Technology: New Generations (ITNG), 2014 11th International Conference on. IEEE, 2014.
[24] Le Thanh, Hieu. "Analysis of Malware Families on Android Mobiles: Detection Characteristics Recognizable by Ordinary Phone Users and How to Fix It." Journal of Information Security 4.04 (2013): 213.
[25] Google Play store [Online]. Available : https://play.google.com/store
[26] Python 2.7 [Online]. Available : https://www.python.org/download/releases/2.7/
[27] Androguard Project [Online]. Available : https://code.google.com/p/androguard/
[28] Android Lollipop 5 [Online]. Available : http://www.android.com/versions/lollipop-5-0/
[29] List of Android Manifest Permissions [Online]. Available : http://developer.android.com/reference/android/Manifest.permission.html
[30] Karegowda, Asha Gowda, A. S. Manjunath, and M. A. Jayaram. "Comparative study of attribute selection using gain ratio and correlation based feature selection." International Journal of Information Technology and Knowledge Management 2.2 (2010): 271-277.
[31] T. M. Cover, J. A. Thomas, Elements of Information Theory, Ed. Wiley, 1991.
[32] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.
[33] Kohavi, Ron. "A study of cross-validation and bootstrap for accuracy estimation and model selection." *Ijcai*. Vol. 14. No. 2. 1995

Ahmed Hesham Mostafa is from Cairo, Egypt. Ahmed was porn in 23-Aug-1990, Ahmed is a 2011 graduate from faculty of computer science, Helwan University with degree in computer science with Excellent and honor degree .Ahmed completed the pre-master program in computer science in 2014.

,He completed the military service in 2013 , He is a Teacher Assistant in computer science department , Helwan University ,He Assisted in teaching many computer science program such as Data structures programming ,Logic Design , Computer architecture ,Algorithms ,Software Engineering , Design pattern In Java and Assembly Programming.

Ahmed interested in Machine Learning, Data Mining, Malware and Mobile and computer security.

Marwa Mohamed Abd El Fatah, Assistant Professor of computer science, Faculty of computers and information, Helwan University .Cairo, Egypt.

She received her B.Sc and MSc. Degree in computer science from Helwan University. Dr. Marwa received the PHD degree in computer science from Helwan University in 2012. Field of interested includes pattern recognition, AI researches and mobile security.

Aliaa A. A. Youssif, Professor of computer science and vice dean for postgraduates and researches at Faculty of computers and information, Helwan University .Cairo, Egypt.

She received her B.Sc and MSc. Degree in telecommunication and electronics engineering from Helwan University. Prof. Aliaa received the PHD degree in computer science from Helwan university in 2000.she was a visiting professor at George Washington University (Washington DC, USA) in 2005. Field of interested includes pattern recognition, AI researches and medical imaging.