Proceedings of the 12th INDIACom; INDIACom-2018; IEEE Conference ID: 42835
2018 5th International Conference on "Computing for Sustainable Global Development", 14th - 16th March, 2018
Bharati Vidyapeeth's Institute of Computer Applications and Management (BVICAM), New Delhi (INDIA)

Deep Learning Based Android Malware Detection Framework

Naman Kapoor
Department of Information Technology
Delhi Technological University
New Delhi, INDIA
Email ID: naman.kapoor12@gmail.com

Soumya Sourav
Department of Electrical Engineering
Delhi Technological University
New Delhi, INDIA
Email ID: s.sourav15@gmail.com

Kapil Sharma
Department of Information Technology
Delhi Technological University
New Delhi, INDIA
Email ID: kapil@ieee.org

***Abstract*— The continuous development in the field of smartphones as well the continuous enlargement of Internet, various softwares are left prone to many malicious activities like Pharming, Phishing, Ransomware, Spam, Spoofing, Spyware, eavesdropping, etc. These threats have not spared the smartphones as well and they are equally prone to these attacks. Our algorithm aims at achieving malware detection with accuracy and efficiency to bolster the faith people show in smartphones while buying them. Our project makes use of Artificial Neural Networks, or broadly speaking, Deep Learning, to perform the required task. ANNs are an amalgamation of accuracy and efficiency, the crux of our project. The accuracy achieved by our model is around 96.75%. Our model runs the test on Android Package Files (APKs) to determine whether a particular application is malicious or not by doing behavior analysis on android application under consideration. This is done by breaking the permissions the app uses and then giving them as an input to the model to classify the apps.**

Keywords*— *Malware Detection, Artificial Neural Network, Permissions, APK

I. INTRODUCTION

[1] The recent ubiquitousness of mobile phones for doing each and every task in day to day life because of their increased functionality has left them vulnerable to many new malware threats. Along with the nature of mobile devices, which are upgrading each day, the mobile threats are also upgrading with each passing day. The term mobile now includes all kinds of devices of IoT. Threats can be in the form of banking trojans, ransomwares, spywares, etc. which often leads to stealing private data and money from the user. Malware detection has continued to be one of the biggest challenges in today's technological world considering the fact that the resources available for doing the same are very limited. [17] According to McAfee malware analysis report 2017 the unprecedented rise of ransomwares is expected to continue in 2017. Google has removed more than 4000 apps from play store in the past year without any information to the users. According to the Telemetry data obtained by McAfee Mobile Threat Research, more than 500,000 mobile users still have these apps installed in their devices thus making them vulnerable to the threats these apps pose. [15] Kaspersky Mobile security products detected 1,319,418 malicious installation packages, 28,796 mobile banking trojans, 2,00,054 mobile ransomwares. TrojanBanker has become one of the most easiest and most used malware by fraudsters.AndroidOS. Asacub is one of the malwares most prominently used. The Q2 2017 results of the Kaspersky Lab were equally threatening with malware detection being at 1,319,148 which is twice the previous quarter's data. In Q2 2017, Adware with a contribution of 13.31%, was the biggest source of malwares. The share increased by 5.99%. The majority of all discovered Installation packages are detected as AdWare.AndroidOS.Ewind.iz and AdWare. AndroidOS. Agent.n. Trojan-SMS malware (6.83%) ranked second in terms of the growth rate: its contribution increased by 2.15 percentage points.

The problem of using deep learning based classifier to detect malwares using android permissions presents 3 main challenges: first, we need to extract an apps main features which will provide the base to classify them; second, important features should be identified and taken into account from the obtained dataset. Third, classification using ANN.

The first problem can be solved using various reverse engineering tools like androguard, apk tool, etc. which extracts an APK file to give its permissions, API calls and other related information about the file. We have used androguard in this project. The second problem is solved using [4] sclearn. ensembl.ExtraTreesClassifier, a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The third problem is solved using Artificial Neural Network model that consists of an eleven-layered structure with 105 number of nodes in each layer.

This paper is organization augments understanding at every level: in Section II, we have discussed the related works in this field; in Section III, we have explained the working of our model; in Section IV, we have presented the components of our methodology; in Section V, we have discussed our experimental results; & finally, in Section VI, we have provided the references of our project .

II. RELATED WORK

The machine learning method for classification [1, 2, 3, 4, 5, and 8] has been very prevalent among data scientists and researchers. With some changes in their framework some encouraging results have been obtained.

The authors [2] used the ensemble features to categorize among the malware apps. They have utilized the androguard as well for breaking down the android permissions along with mining the API calls and software/hardware features. The project is implemented in two phases: first one includes using separate features and the other one using the combined or the ensemble features. The accuracy was around 93% for the ensemble technique.

The authors [4, 9] have used android permissions for their projects as well. However, k-nearest classifier [4] is used in one and random forest tree classifier [9] is used in another. Both are very robust and powerful machine learning algorithms.

By analyzing the number of times each system call has been issued during execution of certain task, Crowdroid [6], a machine learning-based framework, recognizes Trojan-like malware. The trojanized version of an application is very different from its genuine application, since every time it is used, it issues different types of and different number of API calls.

The project [7] makes use of parallel functioning of machine algorithms for storing different features. The algorithms used are Rule Based Classifier, Function Based Classifier, Tree Based Classifier, and Probabilistic Classifier. All the features from these algorithms, after they are trained, are combined and a net result is obtained whether an app is malicious or not. The proposed method generates some good results, however the inclusion of four different algorithms surely speeds it down.

III. METHODOLGY

A. Phase I

First phase of the project involves reading dataset, which includes permissions and API calls of various apps which are classified whether they are malwares or benign applications. Relevant are then selected using a wrapper classifier by segregating features w.r.t. their importance. Irrelevant features are then removed for better generalization and better accuracy as well as reduced time involved in classification.

B. Phase II

Second phase involves training the deep learning model using this dataset and saving weights and the trained model. The artificial neural network is trained on the dataset after the ExtraTreeClassifier segregates the useful permissions. This dataset is dynamic and keeps on changing whenever the program is run. The dataset consists of around 35-40 permissions belonging to the original dataset with almost 330 permissions and generated on 398 applications both benign and malicious. This model after being trained, saves the weights and the model. These weights and model are then called which breaks individual apps to be checked and then the neural network decides whether the app is malicious or benign.

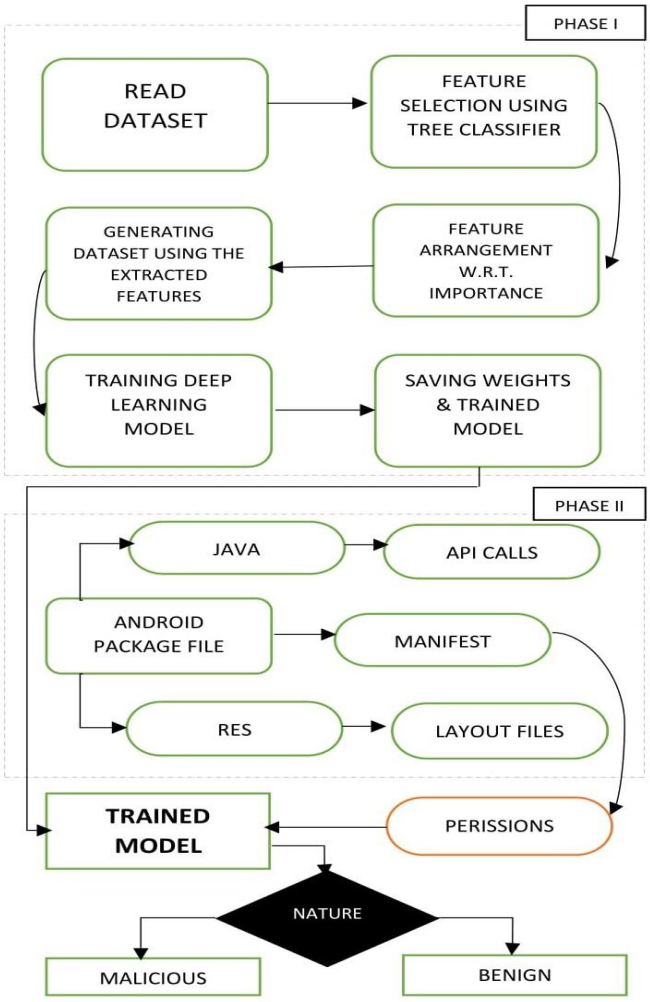

Fig. 1. Work Flow

IV. COMPONENTS OF METHODOLOGY

A. DATASET

Imbalance problem-the situation when the contents of each class are not proportionate- is a very important aspect when using binary classifiers for the detection of malicious code. In our case the absence of malwares and legitimate files in equal proportions causes the imbalance problem. It is need of hour to continuously update the training set with new malicious files for making the classifier better. This is really an important aspect in maintaining such a framework.

The dataset used for analysis is provided by Kaggle, a competitive platform where data miners and scientists can obtain numerous datasets and can compete in various challenges hosted by this domain only. Generally companies and researchers post some data on it and data miners compete to provide the best predicting accuracy on that data. The dataset contains features of 338 applications which are labelled as 'benign' or 'malicious'. This label is used to do feature

 ISSN 0973-7529; ISBN 978-93-80544-28-1

selection and classification analysis by supervised learning. The dataset comprises of a .csv file. The top row contains all the permissions. The presence of a specific permission in an application sample is indicated by 1 and 0 if it is not present.
The dataset is updated to accommodate for new samples using androguard, an opensource project to extract features from APKs.

B. FEATURE CATEGORIES

A **feature** is a significant estimable property or characteristic of an event under observation. Each attribute in the dataset set signifies a unique feature and each entity signifies a data sample (an android application in this case). A feature in this project corresponds to a permission specified by an application. The dataset contains 330 features. The feature vector contains feature data in binary form where '1' indicates the presence of a permission and '0' indicates absence.
The malware analysis approach implemented in this paper uses static android features extracted from the android app which are used to determine if app contains any malicious permissions which depends on a trained classification model. Androguard dose the job of extracting features to train the ANN model combination of malware and benign APKs.
Three categories of features are used for learning phase: 1. App Permissions 2. API calls 3. Standard OS and framework commands.
API calls are the calls which are made to the server in the name of an application using a SDK or an API. Android permissions are requests or permissions acquired by the apps in order to use certain system data and features to maintain security for the system and user.

TABLE I. OVERVIEW OF THE FEATURES EXTRACTED FROM THE APPS.

TYPE	FEATURES
API call related	abortBroadcast; getDeviceId; getSubscriberId; getCallState;getSimSerialNumber ; android.provider.Contacts; android.provider.ContactsContract; ; HttpUriRequest; SMSReceiver; bindService; onActivityResult; Ljavax_crypto_spec_Secret; DexClassLoader; getNetworkOperator; getSimOperator
Command related	. apk; pmsetInstallLocation; pminstall; GET_META_DATA; GET_RECEIVERS; GET_SERVICES; GET_SIGNATURES; GET_PERMISSIONS
Permissions	android.permission.UPDATE_LOCK; android.permission.USER_ACTIVITY; android.permission.VIBRATE; android.permission.WAKE_LOCK; android.permission.WRITE_CALENDAR; android.permission.WRITE_CALL_LOG; android.permission.WRITE_CONTACTS; android.permission.ACCESS_FINE_LOCATION

[a.] A TOTAL OF 330 PERMISSIONS ARE USED.

C. FEATURE EXTRACTION

APK is an archived version of android apps, written in java, along with data and resource files.
An APK is made up of 1). META-INF which holds SHA-I digest, the app certificate, etc. 2). LIB: it contains the compiled code which can be specific to a particular software layer of a processor 3). resources.arsc: it is a file which contains precompiled resources, such as binary XML 4). Res: it contains resources not complied within resources.arsc 5). AndroidManifest.xml: file describing the name, version, access rights, and referenced library files for the application. This file is generally in the form of Android binary XML and which can be converted into human-readable plaintext XML like Androguard, APKTool etc. 6). classes.dex: These are the classes compiled in the dex file format and which are understandable by the Dalvik virtual machine by the Android Runtime. 7). assets: it basically contains android applications assets.
APK files are analysed using androlyze.py. The output consists of 3 objects – 1. An APK object 2. A DalvikVMFormat object 3. An Analysis object.
APK object is a storehouse containing information about the particular APK like its package file, permissions, AndroidManifest.xml etc.
DEX file contains important information like the classes, methods or strings. The DalvikVMFormat corresponds to this DEX file only.
To link the information within classes. dex, Analysis object has some inbuilt special classes.
Reverse engineering is applied to extract features using APKTools or Androguard. Mining the classes. dex further results in API features.

D. PRE-PROCESSING PHASE

In this phase only, those features which are necessary and optimize the generalizability of the prediction algorithm are used. More number of features affects:

- The time spent in classification.
- Problems like over fitting.

Huge and complex datasets affect training models by reducing the efficiency of the classification problem by using unnecessary number of features resulting in increased computation time.
Over fitting is an issue where the model instead of describing the underlying relationship, describes noise and random error. An over fitted model tends to perform very poorly on predictive tasks as it overreacts to minor fluctuations in training data. This is a very common problem with every deep learning model whenever the model tends to get deeper.
Therefore, an efficient dataset is required for more efficient classification and prediction.

E. FEATURE SELECTION TECHNIQUES

Selecting relevant features for using them in model construction is termed as feature selection. It is very beneficial as:

- Models tend to become a lot simple and easier to understand.
- Training time reduces considerably.

- Over fitting is greatly reduced and hence an improvement in generalization is achieved.

Feature selection techniques removes redundant or irrelevant features without incurring any loss in information.

A feature selection algorithm is an amalgamation of a search technique, which selects the features, and an evaluation measure, used to score the feature subsets. The algorithm varies in complexity with some being as simple as testing all the possible subsets of selected features and then winnowing the one with minimal error rate. One of the most influential factor for the algorithm is the choice of evaluation metric and these evaluation metrics are the one's which create a distinction among three main feature selection algorithm: filters, embedded methods and wrappers.

```
Researching important feature based on 330 total features

43 features identified as important:
 1. feature android.permission.READ_PRIVILEGED_PHONE_STATE (0.253527)
 2. feature android.permission.READ_SYNC_SETTINGS (0.110735)
 3. feature android.permission.ACCESS_PDB_STATE (0.060760)
 4. feature android.permission.CLEAR_APP_CACHE (0.057530)
 5. feature android.permission.SET_ACTIVITY_WATCHER (0.042554)
 6. feature android.permission.ACCESS_MTP (0.042409)
 7. feature android.permission.KILL_BACKGROUND_PROCESSES (0.042303)
 8. feature android.permission.RECEIVE_EMERGENCY_BROADCAST (0.032433)
 9. feature android.permission.INTERACT_ACROSS_USERS_FULL (0.024536)
 10. feature android.permission.GLOBAL_SEARCH (0.017994)
 11. feature android.permission.WRITE_CALL_LOG (0.017778)
 12. feature android.permission.WRITE_CALENDAR (0.017599)
 13. feature android.permission.ACCESS_DOWNLOAD_MANAGER (0.017539)
 14. feature android.permission.WRITE_MEDIA_STORAGE (0.016910)
 15. feature android.permission.WRITE_SYNC_SETTINGS (0.014990)
 16. feature android.permission.CAMERA (0.014887)
 17. feature android.permission.ACCOUNT_MANAGER (0.014399)
 18. feature android.permission.READ_EXTERNAL_STORAGE (0.013924)
 19. feature android.permission.DOWNLOAD_WITHOUT_NOTIFICATION (0.011205)
 20. feature com.android.cts.intent.sender.permission.SAMPLE (0.010968)
 21. feature android.permission.ACCESS_INPUT_FLINGER (0.010792)
```

Fig. 2. Important Features with Feature Importance

Our model makes use of wrapper method to figure pout the accuracy of the selected features' subsets. By counting the number of mistakes our model makes while predicting on the hold-out set, the accuracy can easily be determined. Wrapper methods tend to be very intensive when it comes to computations. However the best results on a model can be obtained by this method only.

In order to implement this wrapper sklearn.ensemble.Extra TreeClassifier python library is used. This particular class is very powerful as it uses averaging to improve the predictive accuracy of the subsets obtained by fitting a meta estimator on a number of randomized trees. It also controls over-fitting.

F. CLASSIFICATION

Our main focus towards doing this project was to separate out unknown malicious APKs. Basically there are two types of classification techniques: dynamic and static. When the system is operated at run time for operations like making calls, accessing networks or files, etc., dynamic analysis is performed for detecting malwares. On the other hand, when the information about the program' explicit or implicit observations is obtained from its source code so that classification can be done, it is known as static analysis. Since the program need not be run for classification, static analysis provides very rapid detection. Static analysis makes use of signature based methods. Thus, for the identification an unknown malware before it is executed, generalization of algorithms is must.

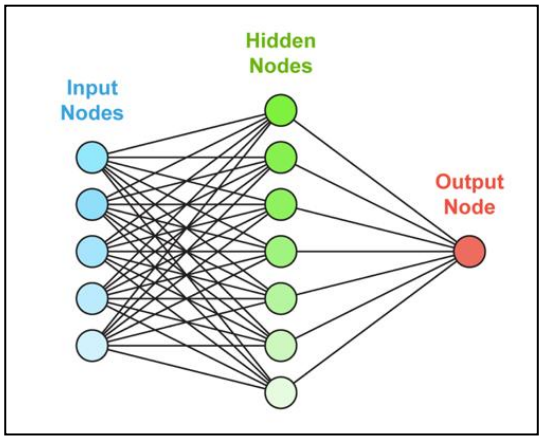

Fig. 3. Basic artificial neural network structure

The problem of classification could be countered with many machine learning algorithms like SVM or regression. However, the main advantage Neural Networks or the multilayer perceptron hold over these algorithms are the fixed number of layer size. The neural networks are parametric models while the machine learning algorithms are not i.e. in neural networks, there are a number of hidden layers depending upon the number of features. The outputs of neural networks can be multiple and they were tested to be faster than the machine learning algorithms.

Artificial Neural Network are basically the algorithms which try to mimic the human brain with input nodes acting as the dendrites, the hidden nodes as axons and the output node as axon terminal.

Each layer has an activation function of itself. For the hidden layers we use the linear rectification function or the ReLU activation function. The output node has sigmoid function as the activation function. Where X is the input matrix and Θ is the activation matrix. The cost function of the neural network is given as:

$$-\frac{1}{m}[\sum_{i=1}^{m}\sum_{k=1}^{k} yk\,(i) * \mathbf{log}(h(x(i)) + (1 - yk(i) * \mathbf{log}(1 - h(x(i)))]$$

The layers can be activated either by random activation or by predefined values.

The classification part of the project makes use of Artificial Neural Network to achieve the task. The model consists of an even-layered structure with 105 number of nodes in each layer. Overfitting issue is dealt with adding dropout layer after every third layer with rate of dropout being 20%.

The model is trained on the winnowed dataset obtained after feature selection. The trained model and the weights are stored for further usage. The model is called in the checkapk.py program where the input is the app to be checked. The weights

are added in the model and the model then looks for the susceptible permissions within the app. Once checked, the model returns a predicted value of the nature of the app.

V.RESULT

We tested our algorithm on different apps both benign and malicious and results were very satisfactory. The androguard easily broke the applications' permissions and the artificial neural network compared the broken permissions with the trained weights. The results were generated with an accuracy of 95.28% with number of epochs being 500.

Basic terms related to confusion matrix:

- True Positives (TP**):** These are the cases which are correctly predicted a 'yes'
- True Negatives (TN): These are the cases which are correctly predicted a 'no'
- False Positives (FP): These are the cases which are wrongly predicted as a 'yes'
- False Negatives (FN): These are the cases which are wrongly predicted a 'no'

Confusion matrix parameters obtained for 500 epochs and 100 nodes

Confusion matrix: [30 5
2 43]Accuracy = (TP+TN)/(TP+TN+FP+FN)
= 73/80
= 93.57%

Recall = TP/(FN+TP)
= 43/45
= 95.55%

Specificity = TN/(FP+TN)
= 30/35
= 85.7%

Precision = TP/(TP+FP)
= 43/48
= 89.3%

F1 score = 2*TP/ (2*TP+FP+FN)
= 2*43/ (2*43+5+2)
= 86/93
=92.4%

TABLE II. CONFUSION MATRIX PARAMETERS VS VARIATION OF NODES

Nodes	Accuracy	Recall	Precision	Specificity	F1 Score
100	91.25	95.55	89.3	86.3	92.47
200	91.35	95.55	88.57	91.5	93.47
300	93.15	95.55	89.3	86.3	92.47
400	91.46	95.55	86.48	89.58	92.47
500	91.35	95.55	88.57	91.5	93.47

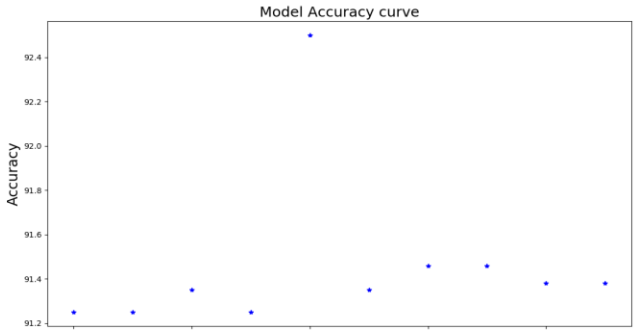

Fig. 4. Model Accuracy Curve

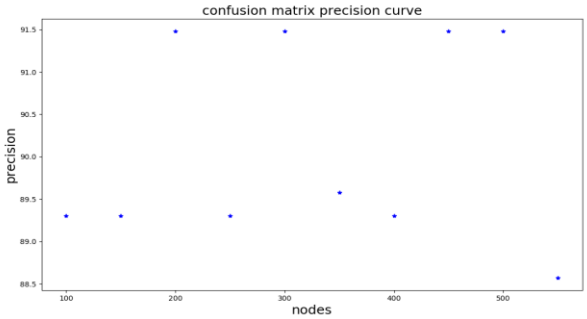

Fig. 5. Confusion Matrix Precision Curve

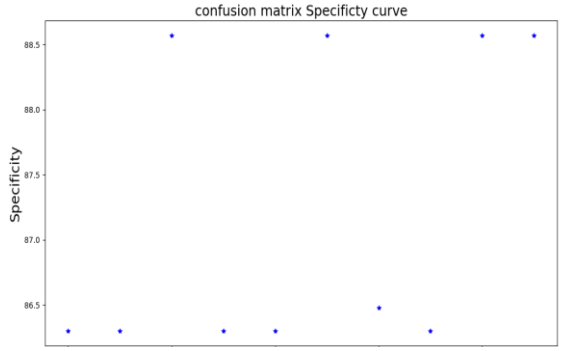

Fig. 6. Confusion Matrix Specificity Curve

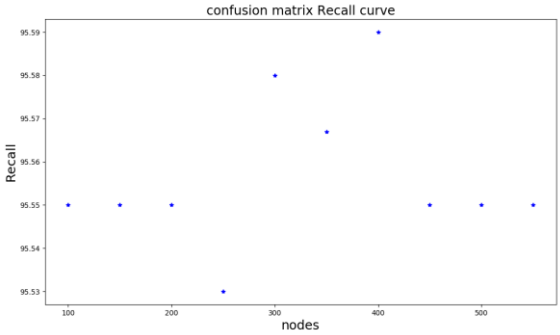

Fig. 7. Confusion Matrix Recall Curve

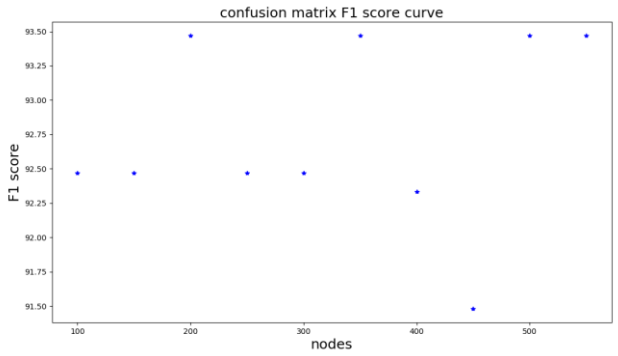

Fig. 8. Confusion Matrix F1 Score Curve

The above curves help us to conclude that for training our model at optimum level we need around 300 nodes in the hidden layers with 500 epochs. the accuracy at node number 300 is highest at 91.35%.

REFERENCES

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

[2] A.M. Ashwini, P. Vinod "Android malware analysis using ensemble features", Springer international publishing Switzerland 2014

[3] Nikola Milosevic, Ali Dehghantanha, Kim-Kwang Raymond Choo "Machine learning aided Android malware classification", Elsevier ltd, 2017

[4] Naser Peiravian, Xing Quan Zhu "Machine learning for android malware detection using permission and api calls", ieee 25th international conference on tools with artificial intelligence 2013

[5] Burguera, U.Z., Nadijm-Tehrani "Crowdroid: Behavior- Based Malware Detection System for Android", SPSM'11, ACM(October 2011)

[6] Suleiman Y. Yerima, Sakir Sezer, Igor Muttik "Android Malware Detection Using Parallel Machine Learning Classifiers",8th International Conference on Next Generation Mobile Applications, Services and Technologies, (NGMAST 2014), 10-14 Sept., 2014

[7] Zami Aung, Win Zaw "Permission-based android malware detection", international journal of scientific and technology research, vol. 2, issue 3 , 2013

[8] Benjamin VA, Chen H "Machine learning for attack vector identification in malicious source code", 2013 IEEE international conference on intelligence and security informatics (ISI). IEEE; 2013. p. 21–3

[9] Boxall A. The number of smartphone users in the world is expected to reach a giant 6.1 billion by 2020; 2015 http://www.digitaltrends.com/mobile/ smartphone- users- number- 6- 1- billion- by-2020/

[10] Schmidt A-D, Clausen JH ,Camtepe A ,Albayrak S . "Detecting symbian os malware through static function call analysis", 4th international conference on malicious and unwanted software (MALWARE), 2009. IEEE; 2009. p. 15–22

[11] Alam MS, Vuong ST "Random forest classification for detecting android malware", 2013 IEEE and internet of things (iThings/CPSCom), IEEE international conference on and IEEE cyber, physical and social computing, green computing and communications (GreenCom). IEEE; 2013. p. 663–9

[12] Introduction to machine learning by Andrew NG. www.coursera.org

[13] Kaspersky Mobile Malware Analysis Report Q2 2017 https://securelist.com/it-threat-evolution-q2-2017-statistics/79432/

[14] Permission based dataset for android malware detection www.kaggle.com

[15] McAfee Mobile threat Report 2017 https://www.mcafee.com/us/resources/reports/rp-mobile-threat-report-2017.pdf

[16] Saba Arshad, Abid Khan, Munam Ali Shah, Mansoor Ahmed " Android Malware Detection & Protection: A Survey", International Journal of Advanced Computer Science and Application Vol. 7, No.2, 2016

[17] Ankita Kapratwar "static and Dynamic Analysis for Android Malware Detection", San Jose State University Scholar Works-spring 2016

[18] Pallavi Kaushik, Amit Jain " Malware Detection Techniques in Android", International Journal of Computer Applications-July 2015

[19] Xiaolei Wang, Yuexiang Yang, Yingzhi Zeng "Accurate mobile malware detection and classification in the cloud", SpringerPlus-October 2015

[20] Pham Thanh Giang "Permission Analysis for Android Malware Detection", The proceedings of the 7th VAST-AIST WORKSHOP Research Collaboration Review and Perspective-November 2015

[21] Fan Yuhui, Xu Ning "The Analysis of Android Malware Behaviours", International Journal of Security and Applications-2015

