Proceedings of the International Conference on Intelligent Computing and Control Systems (ICICCS 2019)
IEEE Xplore Part Number: CFP19K34-ART; ISBN: 978-1-5386-8113-8

Static Analysis of Android Malware Detection using Deep Learning

Sandeep HR [1]
Dept. of Computer Science & Engineering
Amrita School of Engineering, Bengaluru
Amrita Vishwa Vidyapeetham, India
E-mail: sandeephr68@gmail.com [1]

***Abstract*—Android Malware is very common these days as applications are not created by trusted sources. People enter their personal data, save cards and much more, thinking these apps are going to keep them fit or help remind them to do certain essential works which we tend to forget in this busy routine of life. In such cases, detecting the malware before even installing an application would be of great help to us. It could possibly even stop a few crimes. In this paper, we propose to use the fully connected deep learning model for detection of Android malware. Key features of the proposed work include detection of Android malware even before installation, the name of the Android malware, version packages with proven extremely high accuracy of about 94.65%. This model also learns all features from all combinations of features. It includes extensive research and testing to achieve very high accuracy.**

***Keywords*—Android Malware Classification; Malware; Machine learning; Security; Android; Permissions; APK (application package); Deep learning; Data Mining; Data Extraction; Preprocessing; Vector Representation; Behavioral Analysis; Keras; Deep Learning Dense Model; Random Forest Classifier; Virus Share**

I. INTRODUCTION

Cell phones and tablets have turned out to be greatly prominent in the most recent years. Towards the finish of 2016, the number of dynamic cell phones worldwide was right around 7 billion, and in created countries, the proportion between cell phones and individuals is assessed as 120.8%. Given their huge circulation, and furthermore, their abilities, over the most recent two years cell phones have turned into the primary focus for attackers. Android, the open source agent framework (OS) presented by Google, has as of now the biggest piece of the pie, or, in other words, 80%. Because of the receptiveness and prominence, Android [12] is the fundamental focus of strike against cell phones (98.5%), with in excess of 1 million of pernicious applications right now accessible in nature. Malware [14] comprises the principle vector for security strikes against cell phones. Camouflaged as would be expected and valuable applications, they cover up tricky code which performs activities out of sight that undermines the client protection, the gadget respectability, or even client's credit. Some regular models of strikes performed by Android malicious applications are taking contacts, login certifications, instant messages, or harmful buying in the client to expensive premium administrations. Moreover, all this rowdiness can be performed on Android gadgets without the client taking note. An essential worry on the developing Android stage is malware location. Malware recognition strategies on the Android stage are like systems utilized on any stage. Identification is generally broken into a static investigation, by examining an ordered document; dynamic examination, by breaking down the runtime conduct, for example, battery, memory, and system usage of the gadget; or half and half examination, by joining static and dynamic strategies.

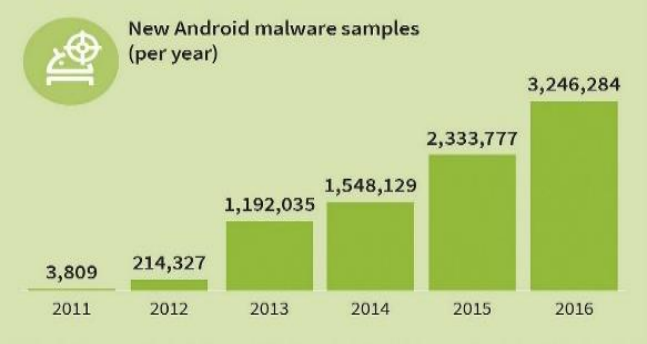

Fig.1 Malware samples per year [11]

The developing need is to identify this Malware for a versatile stage. The general focal point of the examination is to distinguish malware by performing static investigation dependent on a new Android malware informational index. With the expansion of Android applications, experiences. To maintain security honesty and keep up client certainty, different methodologies have been contemplated in the field of malware identification. As malware turn out to be increasingly competent at concealing its malignant purpose using code jumbling, it winds up basic for malware location systems to stay aware of the pace of malware changes. With the far-reaching utilization of cell phones, the quantity of malevolent has been increasingly expanding. Among keen gadgets, they are very much focused on gadgets by malevolent applications due to their popularity. Likewise, because of the consistent development of malignant practices, it is difficult to extricate includes by arduous human component designing and stay aware of the speed of malware advancement. Malware focusing on cell phones is an unavoidable issue in the current life. The discovery of malware is basically a product classification issue dependent on data accumulated from program examination. Identifying new generation malware is an ever-present challenge. In 2017, Kaspersky Lab detected 5,730,916 malicious installation packages, 94,368 mobile banking Trojans and 544,107 mobile ransomware Trojans.

II. RELATED WORK

DeepRefiner performs better than Storm Droid and anti-virus scanners [1]. Also, check for robustness of DeepRefiner Vs. the traditional techniques. DeepRefiner

detects obfuscated malicious applications [1]. In [2] deep-learning-based method is used to detect the families of malicious applications, where results show that engine can detect 97% of the malware at 0.1% false positive rate (FPR) when detecting the fine-grained malware families [2]. Machine learning classifiers in the cloud for real-time mobile malware detection is still to be implemented [3]. [4] Proposes an Android malware detecting system that provides a highly accurate classification as well as efficient sensitive data transmission analysis. Semantic-based features like data dependency graphs and control flow graphs to classify Android malware are to be used further in future [4]. DroidDeepLearner outperforms the existing and widely used malware detection approaches [5]. Fuzzy association rule mining is the future research because the test dataset does not the 168 features [5]. This was the first attempt to construct DNNs for Android application/malware classification [6]. Results showed that DNNs outperformed the age-old methods based on n-grams, CNN proved to be better than LSTM [6]. It demonstrates the effectiveness of DNNs for classification of Android apps [6]. If you fuse information from different parts of an app, a much better and reliable software classifier is constructed [6]. The aim is to design deep neural networks which accept multiple input channels from other sources in the APK file and fuses the information [6]. [7] Utilizes minimum required computational power and resources, to identify Android malware. [7] It is a bio-inspired Hybrid Intelligent Method for Detecting Android Malware (HIMDAM). [7] This approach performs classification by employing Extreme Learning Machines (ELM) to correctly label the malware applications. [7] Also, Evolving Spiking Neural Networks (eSNNs) is used to increase the accuracy and generalization of the entire model. Improving the efficiency of biologically realistic ANN for pattern recognition is the future aim and evaluate with ROC analysis to achieve minimum processing time [7]. An online deep-learning-based Android malware detection engine (Droid Detector) is implemented which automatically detects whether an app is a malware or not resulting with a 96.76% detection accuracy, which does better than the age-old machine learning techniques [8]. In this paper [9] a novel Android malware detection framework is proposed that utilizes many static features to reflect properties of applications in different fields. [9] Totally 7 kinds of the feature are extracted by analyzing files like manifest file, a dex file, and a .so file from an APK file, and these features enrich the extracted information to express applications' characteristics. Droid Deep fuses the concepts of static analysis and deep learning, which makes it easier to stay up-to-date with the malware evolution [10]. In this work, first 32,247 features from 5 main static feature types of Android apps was extracted [10]. After which a Droid Deep is designed on a DBN-based deep learning model to learn features from Android benign app and malware, and classify Android benign apps and malware using the SVM algorithm [10].

III. IMPLEMENTATION

Several packages and functions have been used in this project. Initially, there must be both Apk files of both malware and benign, benign samples can be collected from Google play store and various platforms whereas malware dataset has been collected from virsushare. After obtaining both data samples, data extraction should be performed in order to obtain permissions and those permissions need to be converted to the vector from. After conversion, a model has to be built in such a way that it learns the features or permissions that usually malware Apk files ask for and permissions that normal apps ask for. In other to achieve this, the model has been designed in such a way that it implements a dense/fully connected layer which helps to achieve better results. Since classification has to be made between malware and benign, random forest classifier has been used which is considered as a better classifier of the dataset. Finally, the model detects which is the malware app and which is a benign app with better accuracy.

A. Exploratory Data Analysis:

Exploratory Data Analysis (EDA) gives different information from the data from description to give insights into the data. EDA includes column description, checking for the missing and misleading data, and Graphical Analysis using Visualization. Missing data like having no values or having the data which is irrelevant and NA (Not Available) in some places. Misleading data are like those having datatype (or) the format of the feature different from what is required. Replacing missing data has been done with mode of the feature if it is categorical data and median of the feature with continuous data. In continuous data, if the data has outlier, the mean will be huge and if the outlier is removed it may give very less mean. For that reason, replacement of NA's (Not Available) of continuous data using median has been done.

B. Feature Selection

Feature selection is one of the most important parts of building any model. The modelling of the data completely depends on feature selection. In other words, how strong one attribute implies the other is completely based on the available data. We do feature selection to train the algorithm faster. It reduces the complexity of the model and makes it easier to interpret hence reducing the overfitting problem. Feature selection improves the accuracy of the model if we choose the right subset. We have chosen random forest for our feature selection as with such a selection better results were obtained as when compared with the categorical data. Random Forest is a machine learning method capable of performing both regression and classification tasks. It also undertakes dimensional reduction methods, outlier values, treats missing values and other essentials to do a fairly good job. In random Forest, we grow multiple tresses. To classify a new object based on attributes, each tree gives a classification and we say 'tree votes for that class'. The forest chooses the classification having voted and in case of regression, it takes an average of outputs by different trees. The random forest works with a number of cases in the training set, N. Then a sample of N cases are taken in random with replacement. This sample will be the training set for growing the tree. Each tree is grown to the largest extent possible and there is no pruning. Then we predict the new data by aggregating the predictions of the trees i.e.,

majority votes for classification and average for regression. One of the benefits of identifying the random forest which excites me most is, to identify the most significant variables so that it is considered as one of the dimensionality reduction methods. Further, it outputs the importance of the variable, which can be a very handy feature.

C. Dataset

The dataset has 331 features and a classifier label. Most of the features have binary and multi-labels in the categorical data. The dataset includes information about permissions in API. These are the permissions which are misused by hackers to perform wrong tasks such as to hack the cell phones containing confidential data. A part of the dataset is shown in below fig.

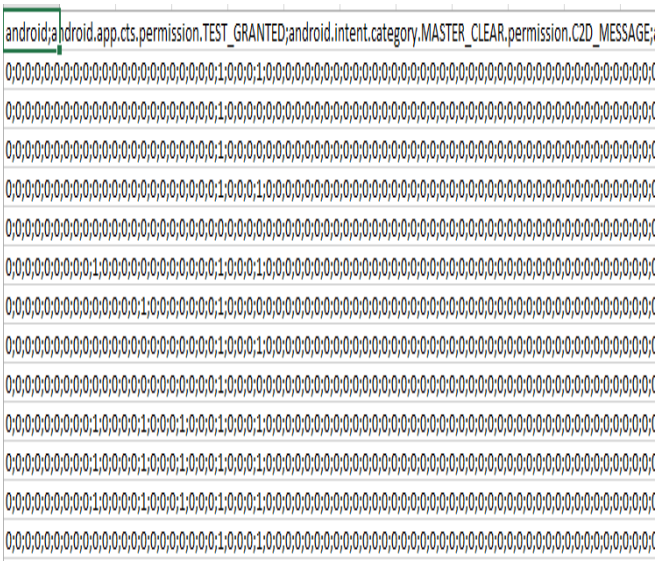

Fig. 2. Dataset Screenshot

Fig 2, shows exactly how the data looks like and what is the format of the dataset used. It's a binary data i.e. either 0 or 1 for all the features. Features are nothing but the permissions that any particular application requests before installation. Some of the permissions are: -
ACCESS_WIFI_STATE,
READ_SYNC_SETTINGS,
READ_EXTERNAL_STORAGE and
WRITE_EXTERNAL_STORAGE.
These are some of the permissions from the dataset which requests the user to write their storage, access Wi-Fi and reading storage. Similarly, there 331 permissions using which the Apk is detected to be a malware or a good ware. We have collected all the necessary permissions that android applications required from cell phones in order to implement our model. These features include both malware and goodware type of permissions which has been classified in the dataset.

IV. TESTING

The data has been validated by using Random forest classifier. The model has been built on all the features. Later, we analyzed data for 400 API permissions which almost include all the API permissions that an app can have. Testing these applications have been done using androguard which is an APK analyzer and works only on Apk files. The malware apps are tested and downloaded from virus-share which is a repository for malware samples. Earlier the data had very few features i.e. fewer API permissions and the results were not accurate. After adding features where it almost covered all the API permissions the results were accurate. The number of neurons in our fully connected layer also played an important role in getting better predictions. On increasing it to 105 neurons per layer the prediction level has increased. Dropout layer is the most important layer in a fully connected layer. Here in our model the dropout has been set to 0.3. The optimizer such as Adam, SGD and activation such as Relu, Tanh and sigmoid vary the final accuracy and prediction. Overall the entire model has been tested with various parameters, optimizers, feature technique and various other layers for the betterment of the model. Some of the test cases are shown below.

Activation function and Optimizer	Accuracy
ReLu, Adam	95.96%
Tanh, Sigmoid	51.6%
Sigmoid, SGD	60.9%

Table 2. Test Cases

Table 2, shows what accuracy changes occurred on varying the Activation function and Optimizers in our deep learning model. The reason the accuracy changes because of their behaviour i.e. Activation function introduces non-linearity and Optimizer minimizes the error and since each Activation function and Optimizers are different from one another the accuracy changes and the result will be better if we use a better combination of Activation function and Optimizers. Some of the model testing is shown below.

Number of features and Inputs (Neurons)	Accuracy
F=less than 100, N= 50	68.75%
F=less than 200, N= 80	79.68%
F=less than 300, N= 100	92.78%
F=greater than 350, N= 105	95.57%

Table 3. Model Testing

Table 3, shows exactly how the accuracy has been increased by introducing more features and inputs (Neurons) that's because the model will learn more details if the features. Similarly, if the number of input neurons is more it helps to obtain more input features at a time and processing time will decrease which will help to achieve better time complexity of the model.

V. EXPERIMENTAL RESULTS

The work proposes the use of a fully connected deep learning model in the detection of Android malware statically. We have implemented using fully connected so that the model will be able to learn all the features from all the combination of features which results in better accuracy. The novelty of this work is that, along with detection of the Android malware, we will be able to identify the malware android application names, version packages and all the required details to say that particular application is a malware or a good ware. Our work proves that feature extraction of Malware application and Goodware application and selection is also a major part in identifying malware applications. A part of the building model is shown in fig. below.

```
 60/318 [====>.........................] - ETA: 0s - loss: 0.2078 - acc: 0.9333
318/318 [==============================] - 0s 218us/step - loss: 0.1851 - acc: 0.9465
Epoch 346/350

 60/318 [====>.........................] - ETA: 0s - loss: 0.1914 - acc: 0.9500
240/318 [=====================>........] - ETA: 0s - loss: 0.2135 - acc: 0.9333
318/318 [==============================] - 0s 492us/step - loss: 0.1854 - acc: 0.9465
Epoch 347/350

 60/318 [====>.........................] - ETA: 0s - loss: 0.1354 - acc: 0.9667
318/318 [==============================] - 0s 126us/step - loss: 0.1809 - acc: 0.9465
Epoch 348/350

 60/318 [====>.........................] - ETA: 0s - loss: 0.2580 - acc: 0.9167
318/318 [==============================] - 0s 114us/step - loss: 0.1861 - acc: 0.9465
Epoch 349/350

 60/318 [====>.........................] - ETA: 0s - loss: 0.1122 - acc: 0.9833
318/318 [==============================] - 0s 114us/step - loss: 0.1878 - acc: 0.9465
Epoch 350/350

 60/318 [====>.........................] - ETA: 0s - loss: 0.2895 - acc: 0.9000
120/318 [==========>...................] - ETA: 0s - loss: 0.2189 - acc: 0.9333
318/318 [==============================] - 0s 697us/step - loss: 0.1779 - acc: 0.9465
False positive rate : 11.428571 %
False negative rate : 6.666667 %
[[31  4]
 [ 3 42]]

Process finished with exit code 0
```

Fig. 3. Building Model

Fig 3 shows how our fully connected layer dense model behaves during the execution time. It clearly shows the number of samples taken per iteration i.e. 60 samples per iteration and in each iteration all the samples are compared with actual and predicted values and respective loss and accuracy has been calculated. Finally, after completion of execution, the final result of False positive and False Negative predations is shown and we use this learnt model to detect malware and good ware Apk files. The proposed work will be useful for all the users as it includes confidential data within their mobile phones. Since our model detects the type of application, i.e., whether the application is malware or good ware before its installation. Results of system testing shows that using deep learning model in malware detection proved to achieve high accuracy of 94.65%. The work can be deployed in identifying all the malware apps and for safer use of the mobile application. Classification of the android application is shown in the fig. below.

```
C:\Users\Dell\Anaconda3\python.exe C:/Users/Dell/PycharmProjects/san/checkapk.py
C:\Users\Dell\Anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion o
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
2018-12-11 10:16:26.131818: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU
2018-12-11 10:16:26.288737: I tensorflow/core/common_runtime/process_util.cc:69] Creating
features: ['android.permission.READ_PRIVILEGED_PHONE_STATE', 'android.permission.ACCESS_PD
['android.permission.INTERNET', 'android.permission.READ_PHONE_STATE', 'android.permission
[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,
not malicious
Successfully done:-

Process finished with exit code 0
```

Fig. 4. Malware Detection

Fig 4, shows how our model will predict a particular Apk given by the user to check if its malware or good ware. It does so by verifying the permissions that the particular application requests. If the Apk requests some of the malware permissions it considers that as malware Apk and the model identifies this since we have already made our model to learn which are good ware and malware Apk permissions.

VI. CONCLUSION

In this paper, we propose a very unique humanoid malware detection framework that uses several options like permissions to mirror the behaviour of the application. Completely different sorts of permissions are extracted from various apk files. Through our projected feature illustration, it's doable to stop feature vector of malware from containing the common properties that seem to be benign applications. Finally, we use the deep learning technique, to identify completely different types of permissions. This design of the model proposed in this paper is appropriate to enhance the malware detection accuracy. The proposed model can observe malware and benign apps by extracting the permissions employed by them and classify them consequently. The accuracy of 94.64% can be considered as a very high rate of accuracy in detection of android malware using our approach because as we know malware applications behave randomly and the permissions vary from one application to another application. In order to find out the most prominent malware features, its important to know what work has not been done previously and what all features does their malware application contain. Here in our work, we have successfully managed to list out what are the features a particular malware or benign application requires during and after execution.

REFERENCES

[1] Xu, Ke, Yingjiu Li, Robert H. Deng, and Kai Chen. "DeepRefiner: Multi-layer Android Malware Detection System Applying Deep Neural Networks." In 2018 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 473-487. IEEE, 2018.

[2] Li, Dongfang, Zhaoguo Wang, and YiboXue. "Fine-grained Android Malware Detection based on Deep Learning." In 2018 IEEE Conference on Communications and Network Security (CNS), pp. 1-2. IEEE, 2018.

[3] Nasrudin, Fairuz Amalina, Ali Feizollah, NorBadrulAnuar, and Abdullah Gani. "Evaluation of machine learning classifiers for mobile malware detection." Soft Computing 20, no. 1 (2016): 343-357.

[4] Wu, Songyang, Pan Wang, Xun Li, and Yong Zhang. "Effective detection of Android malware based on the usage of data flow APIs and machine learning." Information and Software Technology 75 (2016): 17-25.

[5] Wang, Zi, JuecongCai, Sihua Cheng, and Theynjia Li. "DroidDeepLearner: Identifying Android malware using deep learning." In Sarnoff Symposium, 2016 IEEE 37th, pp. 160-165. IEEE, 2016.

[6] Nix, Robin, and Jian Zhang. "Classification of Android apps and malware using deep neural networks." In Neural Networks (IJCNN), 2017 International Joint Conference on, pp. 1871-1878. IEEE, 2017.

[7] Demertzis, Konstantinos, and LazarosIliadis. "Bio-inspired hybrid intelligent method for detecting android malware." In Knowledge, Information and Creativity Support Systems, pp. 289-304. Springer, Cham, 2016.

[8] Yuan, Zhenlong, Yongqiang Lu, and YiboXue. "Droiddetector: Android malware characterization and detection using deep learning." Tsinghua Science and Technology 21, no. 1 (2016): 114-123.

[9] Kim, TaeGuen, BooJoong Kang, Mina Rho, Sakir Sezer, and EulGyuIm. "A Multimodal Deep Learning Method for Android Malware Detection Using Various Features." IEEE Transactions on Information Forensics and Security 14, no. 3 (2019): 773-788.

[10] Su, Xin, Dafang Zhang, Theynjia Li, and Kai Zhao. "A deep learning approach to Android malware feature learning and detection." In Trustcom/BigDataSE/I SPA, 2016 IEEE, pp. 244-251. IEEE, 2016.

[11] www.google.com

[12] R. Vinayakumar, K. P. Soman and P. Poornachandran, "Deep android malware detection and classification," *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Udupi, 2017, pp. 1677-1683.

[13] S. Maniath, A. Ashok, P. Poornachandran, V. G. Sujadevi, A. U. P. Sankar and S. Jan, "Deep learning LSTM based ransomware detection," *2017 Recent Developments in Control, Automation & Power Engineering (RD CAPE)*, Noida, 2017, pp. 442-446.

[14] P. D. Ali and T. G. Kumar, "Malware capturing and detection in dionaea honeypot," *2017 Innovations in Power and Advanced Computing Technologies (i-PACT)*, Vellore, 2017, pp. 1-5. doi: 10.1109/IPACT.2017.8245158