ISSN(Online) : 2456-8910
International Journal of Innovative Research in Applied Sciences and Engineering (IJIRASE)
Volume 4, Issue 3, DOI: 10.29027/IJIRASE.v4.i3.2020.664-668, September 2020

Machine Learning Based Solution for Detecting Malware Android Applications

A.S.Sharan [1], Dr. K.R.Radhika[2]
Department. of ISE, B.M.S College of Engineering, Bengaluru, under VTU, Belagavi, India
Email : sharansrinivas833@gmail.com

Abstract— Smartphone usage has increased rigorously. Android is one of the most used operating systems in various smartphone worldwide. It is open-source and has chances of installing third-party applications without permission. Android is the most vulnerable operating system for a malware attack. This is a big threat to cyber security. In this paper, we make a dynamic analysis using android network traffic logs. We propose an ensemble modelled approach called XGBoost to detect malware and benign applications using the traffic. The proposed model is providing the accuracy of 92.28% and a Kappa coefficient of 0.83. Finally, some of the good set of features from android applications are outlined that helps us to label them as malware and benign. The proposed model is tested across various metrics and they are providing promising results.

Keywords— Android, Malware, Benign, Ensemble Model, XGBoost, Kappa co-efficient, Detection

I. INTRODUCTION

In modern days, smartphones are an essential commodity of life. There many numbers of mobile applications providing various facilities to users. These mobile applications are installed on smartphones. The smartphones contain various numbers of sensors that are used by many of the applications that create a large number of complex data [8,12]. From 2008, an android made its place with the users due to its user-friendly features in applications. Android has access to user's information. The leakage of these data destroys the user's android privacy. Hence attackers are interested in these data. So, they are spreading Malware applications in the android market.

According to the survey made by Kaspersky labs, 80 % of the smartphones uses android as their operating system. One million malware attacks on android devices were recorded in 2019. Malware is the program that disrupts the system operations and stores the users personal and financial information. In the Android platform, to overcome malware application it asks permission from the users while installing [11]. Permissions are offered on its personal because both malicious and benign requires the same set of permissions. Hence this is an ineffective way to detect a malicious application. A Framework must be created to classify malicious and benign applications This can be done by anyone of the two methods.

Static Analysis: It is a quick and inexpensive way to detect malware application by analysing the code. They analyse the variable usage, API calls, code sequences and statements. They can classify the application into malicious or benign without executing the application.

Dynamic Analysis: It is an effective way to detect malicious or benign application by executing the applications in a controlled manner and watching its behaviour. Sandbox is commonly used for dynamic analysis.

We use dynamic analysis because of its efficiency. During the execution of any applications, few features will be recorded. All response and requests are recorded. This log is used to label the application into benign and malware.

A. Working Procedure and Organization of paper

Initially, we shape the dataset and remove all the outliers. Outliers are the numbers which are at the abnormal position other than the normal values. Then the data is normalized and standardized. Now the dataset is divided for training a testing part. The algorithm is trained and then predicted across various algorithms such as K-mean, Decision tree, Naive based, SVM, Ada-boost and XGBoost. By adopting standard metrics such as recall, precision, accuracy, F1 score which is also called as sensitivity and specificity to measure how efficient and to check the performance of our work by running the algorithm on dataset produce by reputed organizations.

In the next portion, we discuss the remaining works of our paper as mentioned below. Section II deals with the bibliography that has been the base papers for the current work carried out in this paper, Section III emphasizes on design, methodology and implementation of the proposed algorithm, Section IV discusses results and analysis that includes metrics and performance evaluation with a different algorithm. Finally, the paper is ended by the conclusion in section V.

II. LITERATURE SURVEY

Anshul Arora [2] et.al. has developed a detector to detect malware and benign applications using the rule-based classifier. Initially, they analyse the features according to the behaviour of network traffic. Then they distinguish the features depending on the importance. They build the classifier and train the traffic. The classifier is used to predict the traffic as malware or benign. This experiment is only specific to those malware which is connected to remote servers in the background.

Westyarian et al. [3] use 205 malware and benign applications for analysis. They use kernel-level logs such as API system calls based on permissions for analysis. They make a correlation comparison that doesn't affect the machine learning algorithm for detecting malware. They classify using SVM, J48 and random forest machine learning algorithms.

Mehedee Zaman et al. [4] has provided in detail a method to detect the malware. They create an application URL table which has 4 processes that are dumping the packets, logs of netstat, important features from packet dump and aggregation of information from packet dump and logs of netstat. Finally, the table consists of application ID and URL for which they contact. Then they monitor the applications which try to contact the malicious domains. They make an analysis of application behaviour using sys-call trace to build a model to detect the malware applications.

Taniya Bhatia [6] et al. has performed a dynamic analysis. They have collected 50 malicious and malware applications. Then they use sys-call capture and collect all the traces of applications. They try to understand the behaviour of this application by considering the system calls made by these applications. Decision tree, J48 and random forest algorithm are used to classify the applications as malware and benign.

Satish Kandukuru et al. [7] has built a hybrid model to classify malware and benign applications. They make a two-level analysis. First, they extract naïve library, byte code and XML file from the application. Then they extract metadata and list of permissions requested by the applications. They analyse using this permission bit vectors. Secondly, they make a traffic analysis of TCP and HTTP packets. They use to try to find out various keywords such as OS details, SIM serial details and so. These keywords give information about data leakage. Some features are extracted from this traffic and analysis is made to classify the application as malware and benign.

Anshul Arora and Sateesh [9] have built an NTPDroid model to detect the malware application. They make both static and dynamic analysis. Initially, they extract features by capturing network traffic. Then they extract permissions from an XML file. The features and permissions are combined to make combine patterns. Finally, they generate patterns of both malicious and benign applications, that helps in defending malware. They classify the application by 3 ways first by using permission alone, second by traffic and third by combining permission and network traffic.

According to Pradeep Kumar Tiwari et al. [13], they collect all the applications. Then each application is subjected to 3 types of analysis such as pre-static, static and dynamic analysis for extracting important features. Then the analysis is monitored in 4 levels, in first they make package level analysis by monitoring applications meta-data and manifest files, second, they make user-level monitoring how the device behaviour changes while interacting with the users, third they make application-level monitoring to check if there is any information leakage, frequency of incoming and outgoing messages, fourth they make kernel-level monitoring by monitoring system calls and inter-process communications. They use Dropbox for extracting features from the traffic. Summarising all these analysis results, then they are labelled as malware and benign applications.

III. METHODOLOGY

This section describes the dataset description, architectural diagram and complete working procedure of the detector.

A. Dataset Description

The dataset contains the collection of 7500 benign applications. This collection is a group of 50 android families. The collection also has 5500 malware applications of 180 families of malware. The traffic is in pcap files. Initially, pcap files are preprocessed in Droid Box to capture traffic for getting network features [5]. The below table gives the feature description of the dataset.

Table 1: Dataset Feature Description

Features	Description
TCP Packets	Total number of TCP packets transmitted and received via the communication
Distinct Port TCP	Total number of packets other than TCP
External IP's	Number of External IP's tried to communicate application
Volume of Bytes	Total Bytes transmitted from application to external site
UDP Packets	Number of UDP Packets send and received via a communication
Source Application Packets	Number of packets transmitted from application to server
Remote Application Packets	Number of packets received from the application to external site
Source Application bytes	Number of bytes transferred between application and server
Source Application bytes	Number of bytes transferred between server and external sites
Duration	Total time of communication
Local Packet Rate	Average packet rate between application and server
Remote Packet Rate	Average packet rate between server and external sites
DNS Query	Number of DNS queries
Type	Target values benign or malware

B. Architectural Diagram and Working

We make a dynamic analysis on Android Network Traffic dataset [5] in our paper. The traffic created by the applications while executing is captured. Initially, we read dataset. It contains various features as discussed in the above section. The target values are either Benign or Malware. There are 5000 benign rows as the target type and 3500 malware rows as their target type. The architectural diagram is shown in figure 1. We clean the data by removing the outliers and Null values. Outliers are the values which lie in some abnormal position than other values. If we do not remove these outliers, then it may affect the model during the time of classification. There are few outliers which are found during the analysis listed below in figure 2.

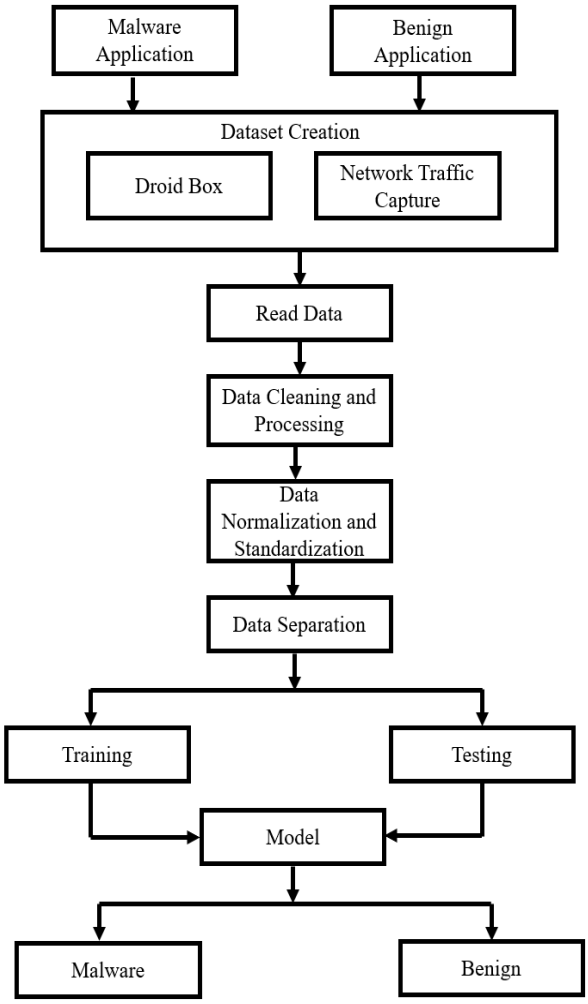
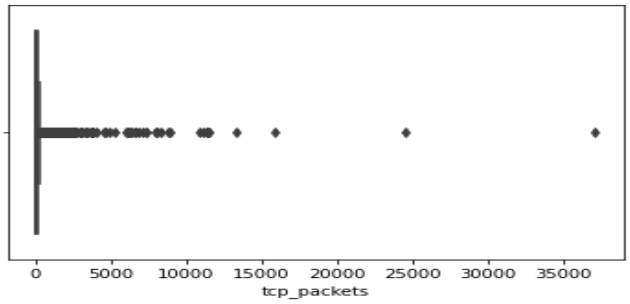
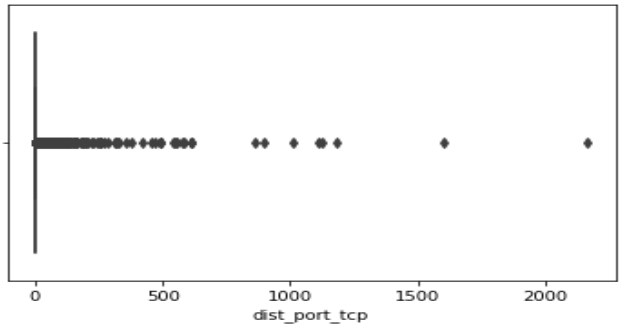


Figure 1: Flowchart of the Working Process



(i)



(ii)

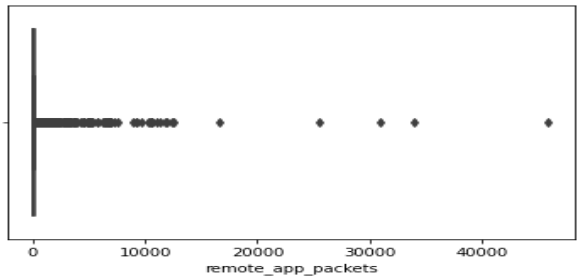


Figure 2: Outliers in (i) TCP Packets (ii) Distinct port TCP (iii) Remote Application Packets

We can observe in figure 2 of (i) where the values in TCP packets are abnormal the values are quite continuous till 15000 then on it suddenly moves to 25000 and then to 37000. Where the values are in abnormal positions. Till 15000 the values are normal then on there is a sudden change. Hence, we choose the data with TCP packet value less than or equal to 15000. Similarly, in distinct port TCP values, less than 1500 and Remote application packets less than 15000 values are normal values others are at an abnormal place. Before removing outliers, we remove the columns which are filled with null values.

Once all the null and abnormal values are removed, we standardize the data using the robust scalar technique. The use of robust scalar is reducing the effect caused by the outliers. Robust scalar removes the median from the feature vector and scale the data according to the range between 1st and 3rd quartile according to the formula listed below.

$$Scaled\ value = \frac{Xi - Median}{Quartile3(x) - Quartile1(x)}$$

After standardizing the values, we separate the data into two parts training and testing. The separation is done in 70-30 pattern i.e. 70% for training and 30% for testing. Then we train these data to various models/algorithms and predict the malware and benign applications. The results of these models are discussed in the next section

IV. Results and Analysis

A. Metrics

As discussed in the previous section the performance of the algorithm is measured by standard metrics figure 3 using standardized formulas such as Precision, Recall, Accuracy, F1 Score are applied [1] part of specificity and sensitivity along with analysis [10]. The above metrics are calculated based on the following outcomes.

		Prediction	
		Malware	**Benign**
Reality	**Malware**	True Negative	False Positive
	Benign	False Negative	True Positive

Figure 3: Confusion Metrics

1. True Positive (TP), when occurrence and classification both are benign.
2. False Negative (FN), when occurrence being positive i.e. benign, but the classification is malware.
3. True Negative (TN), when occurrence and classification both are malware.
4. False Positive (FP), when occurrence being negative i.e. malware but the classification is benign.
5. Recall, the proportion of correctly classified positive occurrences from positive cases.

$$Recall = \frac{TP}{TP + FN}$$

6. Precision, the proportion of correctly classified positive occurrences from cases that are predicted as positive.

$$Precison = \frac{TP}{TP + FP}$$

7. Accuracy is the proportion of correct classifications by a total number of cases.

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

B. Results and Discussion of Analysis

Ensemble models are based on a machine learning approach they combine all the decisions made by multiple models to improve the overall performance. This can be done using advance techniques such as Bagging, Stacking and Boosting. Stacking makes a prediction using multiple models to form a new model. Bagging creates subsets of observation with replacement from the data. The subset must be less than the original sets. Boosting is a sequential process to correct the errors made by the previous model. XGBoost algorithm is an advancement of Gradient Boost algorithm. XGBoost algorithm is based on Bagging and Boosting ensemble techniques. When we apply XGBoost model on our dataset we get the following results as shown in figure 4.

```
XGBoost
0.9227823867262285
              precision    recall  f1-score   support

      benign       0.93      0.94      0.94       942
   malicious       0.91      0.89      0.90       625

    accuracy                           0.92      1567
   macro avg       0.92      0.92      0.92      1567
weighted avg       0.92      0.92      0.92      1567

Accuracy: 92.28%
cohen kappa score
0.8383220563720161

Confusion Matrix
[[889  53]
 [ 68 557]]
```

Figure 4: Result of XGBoost model

We have applied our dataset with various algorithms such as Random Forest, Naive Based, K-neighbors, XGBoost, AdaBoost and SVM with custom kernels linear regression, polynomial, sigmoid and radial bias function. The accuracy with these models is shown below in figure 5. By seeing the figure, we may say that XGBoost model is providing the best accuracy rather than other models and it is the best fit for this dataset. Kappa coefficient is used to check the inter and intra-reliability of the experiment, which should value around 0.8-1 for best fit. And XGBoost algorithm Kappa co-efficient is around 0.83 which has good reliability on the experiment.

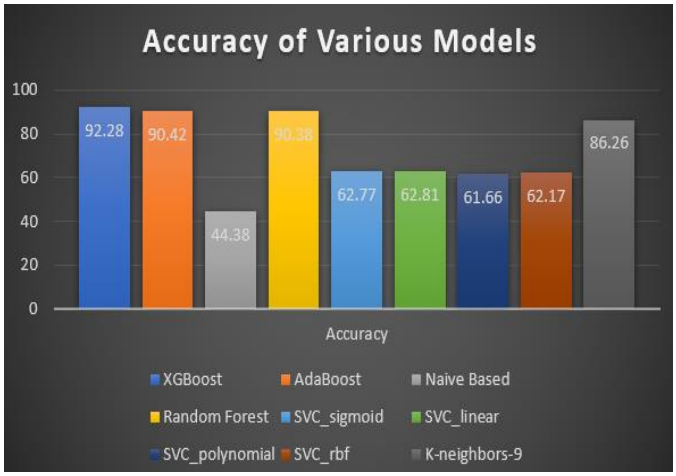


Figure 5:Accuracy of Various Models

V. Conclusion

In this paper, we initially discuss the malware applications in android and its effects. Then we explore various base papers and their implementation to classify malware and benign. Next, we analyze android network traffic dataset and discuss the methodology. Finally, we discuss the results. We have used various models to predict using our dataset. But XGBoost is giving better performance with the accuracy of 92.28% rather than other models and it is the best fit for our dataset. The XGBoost is tested across various metrics and has provided promising results.

References

[1] Powers. David Martin. "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation." (2011).

[2] Arora, Anshul, Shree Garg, and Sateesh K. Peddoju. "Malware detection using network traffic analysis in android based mobile devices." In *2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies*, pp. 66-71. IEEE, 2014.

[3] Rosmansyah, Yusep, and Budiman Dabarsyah. "Malware detection on android smartphones using API class and machine learning." In *2015 International Conference on Electrical Engineering and Informatics (ICEEI)*, pp. 294-297. IEEE, 2015.

[4] Zaman, Mehedee, Tazrian Siddiqui, Mohammad Rakib Amin, and Md Shohrab Hossain. "Malware detection in Android by network traffic analysis." In *2015 international conference on networking systems and security (NSysS)*, pp. 1-5. IEEE, 2015.

[5] Cao, Dong, Shanshan Wang, Qun Li, Zhenxiang Cheny, Oiben Yan, Lizhi Peng, and Bo Yang. "Droidcollector: A high performance framework for high quality android traffic collection." In *2016 IEEE Trustcom/BigDataSE/ISPA*, pp. 1753-1758. IEEE, 2016.

[6] Bhatia, Taniva, and Rishabh Kaushal. "Malware detection in android based on dynamic analysis." In *2017 International Conference on Cyber Security And Protection Of Digital Services (Cyber Security)*, pp. 1-6. IEEE, 2017.

[7] Kandukuru, Satish, and R. M. Sharma. "Android malicious application detection using permission vector and network traffic analysis." In *2017 2nd International Conference for Convergence in Technology (I2CT)*, pp. 1126-1132. IEEE, 2017.

[8] Pang, Ying, Zhenxiang Chen, Xiaomei Li, Shanshan Wang, Chuan Zhao, Lin Wang, Ke Ji, and Zicong Li. "Finding Android malware trace from highly imbalanced network traffic." In *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, vol. 1, pp. 588-595. IEEE, 2017.

[9] Arora, Anshul, and Sateesh K. Peddoju. "NTPDroid: a hybrid android malware detector using network traffic and system permissions." In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pp. 808-813. IEEE, 2018.

[10] Murtaz, Muhammad, Hassan Azwar, Syed Baqir Ali, and Saad Rehman. "A framework for Android Malware detection and classification." In *2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, pp. 1-5. IEEE, 2018.

[11] Taheri, Lava, Andi Fitriah Abdul Kadir, and Arash Habibi Lashkari. "Extensible android malware detection and family classification using network-flows and api-calls." In *2019 International Carnahan Conference on Security Technology (ICCST)*, pp. 1-8. IEEE, 2019.

[12] Sabhadiya, Sagar, Jaydeep Barad, and Jaydeep Gheewala. "Android Malware Detection using Deep Learning." In *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, pp. 1254-1260. IEEE, 2019.

[13] Tiwari, Pradeep Kumar, and T. Velayutham. "Automated Ensembling of Features from Android Applications for Malware Detection." In *2019 International Conference on Cutting-edge Technologies in Engineering (ICon-CuTE)*, pp. 4-8. IEEE, 2019.