International Journal of Future Generation Communication and Networking
Vol. 14, No. 1, (2021), pp. 145–156

Statistical Analysis Of Malware In Anroid With The Techniques Of Machine Learning

Hemant Kumar
Jamia Hamdard
hemant.sscbs@gmail.com

Akshay Chamoli
Jamia Hamdard
akachamoli@gmail.com

Subodh Kuma
Jamia Millia Islamia
subodhkumar588@gmail.com

Abstract

The use of smartphones continues to sour, with Android leading the way. Google claims to have around 1.4 billion active mobile devices across the globe. According to The Telecom Regulatory Authority of India, India has become the second country globally with a user threshold of 1.03 billion users. The massive user base has caught the eyes of cybercriminals trying to attack Android using malware. Malware is malicious software created to destroy computer or electronic systems without the knowledge of the user using the system. This work aim has created an efficient malware detection system with machine learning-enabled features for Android Environment so that they can sense malware applications with the help of static features. The system extracts various permissions and API Tags from the Android applications. It uses these features along with five unique machine learning techniques for classifying whether the application is malicious or benign. The experimental results are promising as we have achieved high accuracy in detecting Android malware with all the classifiers. We have also analyzed the effect of a number of Android applications used in the database on the accuracy of classifiers.

1. Introduction

Android applications are generally used in cell phones as a mobile software application and are created by Google to run on Android platforms like smartphones, tablets, Google TV, and other devices compatible with running Android featured operating system. Android applications utilize advanced hardware and software to bring benefits and values to their users. According to statista.com, the number of accessible applications in the Google Play Store outperformed 1 million applications in July 2013 and was most recently positioned at 2 million applications in February 2016 [1]. Google says there are 1.4 billion active mobile devices [2] worldwide. According to The Telecom Regulatory Authority of India, India has become the second country globally with a user threshold of 1.03 billion users [3]. The Android platform's openness makes it attractive to users, but the freedom or openness users appreciate is also a pitfall of the forum. Cybercriminals nowadays are finding more complex ways to get into android systems and breach the security of the system. Cybercriminals exploit it by posting malware spreading apps in deceitful attempts to steal personal information. In this case user's mobile security is compromised. So the users who download malicious apps unknowingly face a mobile privacy threats. The research carried out by Pulse secure, found that 97% of malware focusses the Android operating system [4].

Malware is malicious software and is developed for damaging the computer system irrespective of the knowledge of the user [5]. Trojan horse, worm, virus, spyware are all classified as malware. It is necessary to analyze malicious apps to understand the risk associated and their intentions. There are various types of analysis, including Static Analysis and Dynamic analysis as the broad domain. Examining malicious software without performing the process is called static analysis whereas, exploration of the infected file during its completion is known as dynamic analysis [5, 6]. It is also called behavioral analysis. In this project, we have used static analysis of malicious apps. The disassembly method is one of the old procedures of static analysis. Machine learning classifiers are used for classifying malicious applications having permissions, API Tags, and a combination of both [7, 8, 9, 10, 11].

Our contribution in the paper are as follows:

ISSN: 2233-7857 IJFGCN

1. Extraction of Android App static features by using Androguard.
2. Classification of Android applications as malicious or benign by using five machine learning classifiers: Naïve Bayes, KNN, Logistic Regression, C4.5, and SVM.
3. Analyze the effect of three sets of static features (permissions only, API Tags only, and permissions and API tags) on the performance of the classifiers.
4. Analyze the effect of the number of Android applications on the performance of classifiers.
5. Based on our database we have also compared the top ten Android app permissions and API tags of both benign and malicious apps and then concluded the dangerous permissions and API Tags.

2. Related Work

In the domain of static analysis for malware detection in Android applications, several studies have already been processed. Sanz et al. [12] introduced PUMA (Permission Usage to detect Malware in Android), a process for sensing malevolent applications using the consent usage of each application. They used machine learning models to perceive whether an application is malicious or not. A dataset of 357 benign and 249 malicious applications was used in the technique, and Android Asset Packaging Tool (AAPT) was used to get permissions from the APK file. Justin Sahs et al. [13] used a machine learning process to detect Android malware. The technique used Androguard (an open-source project) to get features from APKs and then used these features to train a One-Class SVM. The system was limited to just permissions (built-in and non-standard) and CFGs of the input applications. One more approach towards Permission-Based Malware Detection which is proposed by Aung et al. [14] have used K-means clustering, Random Forest DT (decision tree) and CART (Classification and Regression Tree) algorithm. The accuracy rate improved to 91.75% with Random Forest DT and the most efficient orignal positive rate of 97.8% with CART.

Aafer et al. [15] presented DroidAPIMiner: generic data mining technique to develop a classifier for applications having inbuilt android feature. Their procedure had compared performance of four classifiers: ID5, C4.5, KNN and SVM in terms of both the process of various feature extraction i.e. approval based and API calls based feature set with package level and parameter information. The data set taken for this paper consists of 3987 malware samples from Android Malware Genome Project and 500 benign applications from each category in Google Play [16]. They reported KNN with an efficient performance model and has given the accuracy of 99.9%. The false positive rate is very low as 2.2% when around 189 features are taken from a feature set. Akanksha Sharma et al. [17] gave a proactive technique for static malware detection. The features which were removed based on API calls as well as permissions. Correlation based as well as information gain feature selection process and used to select most efficient features. The dataset created from the selected feature set was validated using Naive Bayesian and K-Nearest Neighbor classifiers. They used APKAnalyser for reverse engineering.

3. Proposed Methodology and Implementation

The suggested procedure consists of these four steps. They are:

STEP 1: Creation of an efficient database having a variety of Android malware belonging to different malware families and benign apps from various categories of applications such as communication, education, health etc. We have created 4 sets of databases having a different number of Android apps.

STEP 2: Extracting features such as Android permissions and API tags and creating three sets of features-permissions only, API tags only, and permissions and API tags.

STEP 3: Preprocessing the three sets of features by using the filter to remove unnecessary features.

STEP 4: Building the models of a Machine learning classifier using the 3 sets of features and also by using 4 different databases of apps and subsequently analyzing the performance of various classifiers based on the feature set and database set.

ISSN: 2233-7857 IJFGCN

I. Creation of Database

Here we label the procedure we monitored to get data from the android application file. The basic steps we have followed for each application are:

1. First, we downloaded the android apps of different malware families from AndroMalShare [18] and goodwares from Google Play Store [16] as shown below.

Malware Families	Count	Goodware Families	Count
DroidkungFu	10	Arcade	5
FakeInst	10	Tools	22
Opfake	10	Business	8
Geinimi	10	Communication	10
FakeLogo	10	Education	11
Kmin	10	Entertainment	9
Adrd	10	Lifestyle	10
YZHCSMS	10	Music & Audio	8
PJApps	10	Photography	7
Legacy	10	Puzzle	7
		Shopping	9

2. We decompressed the android apps using Androguard [19] and extracted the selected features like **Permissions requested and API Tags** for each android app by using our python scripts.
3. We build a dataset in a CSV file format with the extracted data.

In step 2, we processed the AndroidManifest.xml file to extract the data.

Using our python scripts we extracted the permissions and created the feature vectors in the following way.
For each Android app, we retrieved the selected features "permissions requested" and "API Tags". The selected features with their values are kept as a binary number (0 or 1) and is signified as a sequence of comma separated values which is also called as CSV format.

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0, MALICIOUS

Figure 1: Feature Vector created using extracted android permissions

0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,BENIGN

Figure 2: Feature Vector created using extracted API Tags

Features

Features used in the dataset are the Permissions requested, API Tags (Tags defined in Androguard[19] corresponding to APIs used in the app), and Permissions requested and API Tags used by the Android Applications[20].

Few samples of permission features are:

- **android.permission.READ_PHONE_STATE**: Permits access in read only mode. This is crucial agreement in order harm an android application
- **android.permission.INTERNET**:

ISSN: 2233-7857 IJFGCN

Permits the applications for opening network sockets for the usage of internet. If any application doesn't require the internet and still it request the INTERNET access then it can be used for malicious purpose

- **android.permission.CALL_PHONE**:
  Permits the application to initiate a phone call without accessing the Dialler user interface for the user to approve the call. Malevolent use of this permission can incur charges to the user.
- **android.permission.ACCESS_COARSE_LOCATION**:
  Permits an app to access approximate location. Attackers can use this permission maliciously to track the location of the user.
- **android.permission.ACCESS_WIFI_STATE**: Permits applications to access information about Wi-Fi networks. By making use of android permissions. INTERNET along with this permission access, attackers gain access to sensitive information.

Few Samples of API Tags are:

- **Android**: It consists of resource classes and is used by the applications which are included in the platform and is used to define application approvals for system features.
- **Bluetooth**: Offers classes that manages Bluetooth functionality, which includes various features such as device scanning, connection establishment among devices for communication and management of data transfer among devices.
- **Location:** Consists of the framework API classes that defines services enabling location based features.
- **Reflection**: brings reflective access to description info.
- **Widget**: Consists of the components essential to create "app widgets", which users use to integrate with other applications (such as the home screen) to speedily access application data and services without starting of a new activity.

II. Preprocessing the Database

In Machine Learning applications, an enormous number of extricated highlights, some of which excess or immaterial, present a few issues, for example, deceiving the learning calculation, over-fitting, lessening over-simplification, and expanding model intricacy and run-time [21,22]. These antagonistic impacts are significantly more critical when applying Machine Learning strategies on cell phones since they are regularly limited by preparing and capacity abilities, just as battery power. Therefore Applying the Remove Useless in weka [13], filtered out the dataset in a preparatory stage and enabled the malware detector to work more proficiently, with a quicker detection period. Yet, decreasing features should only be performed when a high level of accuracy can be preserved.
We have preprocessed the database using "**Remove Useless"**. It is an unsupervised attribute filter used to remove attributes that do not vary at all or that vary too much

III. Building the model using classifiers and detecting the malware

Our job is to develop a model that can classify an app as either malicious or benign. To do that, we have used five different machine learning algorithms for classification Naïve Bayes Classifier, KNN, Logistic Regression, C4.5, and Support Vector Machine. These five different classifiers belong to other classifiers' families, KNN belongs to the lazy classifier, Naïve Bayes. Logistic Regression belongs to the family of Probabilistic classifier, C4.5 belongs to the Decision tree, and The ML technique capable of dividing the training data set by optimal separation is done with the Support Vector Machine technique [24, 25].

We examined two experiments.

First Experiment: In the first experiment, we have extracted around 135 permissions, 41 API tags from 200 apps. We created four databases having 50 instances, 100 instances, 150 instances and 200 instances

ISSN: 2233-7857 IJFGCN

of permissions, API Tags, and permissions plus API Tags mixed respectively. So basically this experiment consists of three parts that are:

1. Permission Based Analysis
2. API Tags Based Analysis
3. Permissions plus API Tags based Analysis

All these three parts consist of these two steps.

STEP -1- Preprocess each database using "Remove Useless" filter in Weka [13].

STEP-2-Classifiy each database using five different machine learning algorithms in Weka [13].

1. Naïve Bayes **2**. KNN**3**. Logistic Regression **4.** C4.5 **5.** SVM

To test our created grouping models, we have utilized cross approval technique with 10 folds, i.e., our dataset is part multiple times into 10 unique sets for learning (90% of the absolute dataset) and testing (10% of the complete information). Note the Accuracy, True Positive Rate, and False Positive Rate of every classifier.

Compare the ACCURACY and TIME EFFICIENCY of each classifier.

Second Experiment: In this experiment, we have extracted top ten permissions and API Tags from the dataset of both benign and malicious apps. We have then compared the top ten permissions and API Tags from both benign and malicious apps and have concluded dangerous permissions and API Tags.

4. Result and Analysis

Results of both the experiments are given below:

1st Experiment Results:
We generate the classification model using five different classifiers described earlier for permissions, API Tags, and permissions plus API Tags as features and run them for four different datasets, a dataset of 50 instances,100 instances, 150 instances and 200 instances respectively. We have used Weka [13] for classification.

Table 1: The result of each classifier on permissions based dataset

Classifier	Dataset of 50 instances		Dataset of 100 instances		Dataset of 150 instances		Dataset of 200 instances	
	Acc.	TPR	Acc.	TPR	Acc.	TPR	Acc.	TPR
Naïve Bayes	90%	0.840	95%	0.900	90.66 %	0.933	92%	0.960
KNN(k=5)	86%	0.760	90%	0.820	89.33 %	0.827	95%	0.950
Logistic Regression	90%	0.880	93%	0.960	95.33 %	0.960	96.5 %	0.960
C4.5	90%	0.880	95%	0.980	93.33 %	0.933	96%	0.970

ISSN: 2233-7857 IJFGCN

SVM	90%	0.880	97%	0.960	98%	0.973	98%	0.980

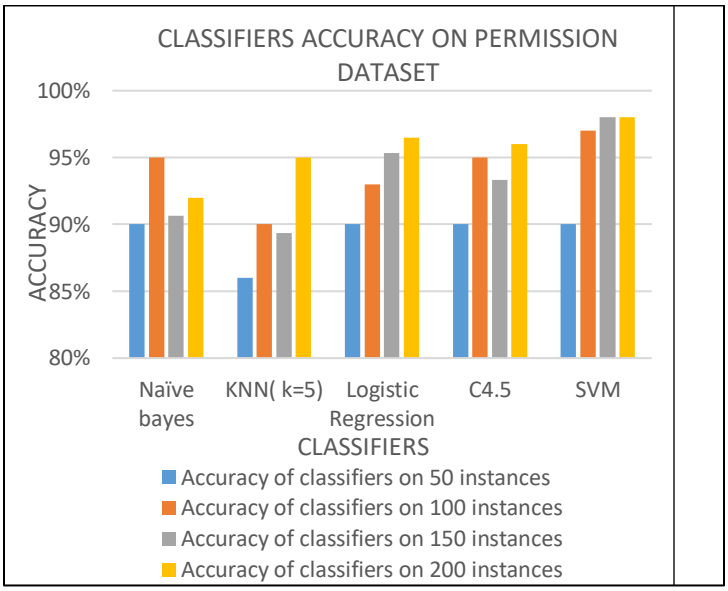

Figure 3: Bar Graph showing classifier wise accuracy for permission based analysis

In permission based analysis we found that the accuracy of SVM is the highest i.e., 98% in the dataset of 200 and 150 instances while KNN holds the lowest accuracy i.e. 86 % in case of 50 instances. Also, we found that in most of the cases, as we increase the number of instances the accuracy of classifier increases with the slight exception in case of Naïve Bayes

Table 2: The result of each classifier on API Tags based dataset

Classifier	**Dataset of 50 instances**		**Dataset of 100 instances**		**Dataset of 150 instances**		**Dataset of 200 instances**	
	Acc.	**TPR**	**Acc.**	**TPR**	**Acc.**	**TPR**	**Acc.**	**TPR**
Naïve Bayes	86%	0.960	93%	0.960	89.33 %	0.933	91%	0.940
KNN(k=5)	86%	1.000	90%	1.000	91.33 %	1.000	93.5 %	1.000
Logistic Regression	82%	0.800	87%	0.880	89.33 %	0.907	92.5 %	0.930
C4.5	84%	0.920	91%	0.980	93.33 %	1.000	93%	0.970

ISSN: 2233-7857 IJFGCN

SVM	86%	0.880	92%	0.980	94%	0.973	95.5 %	0.98 0

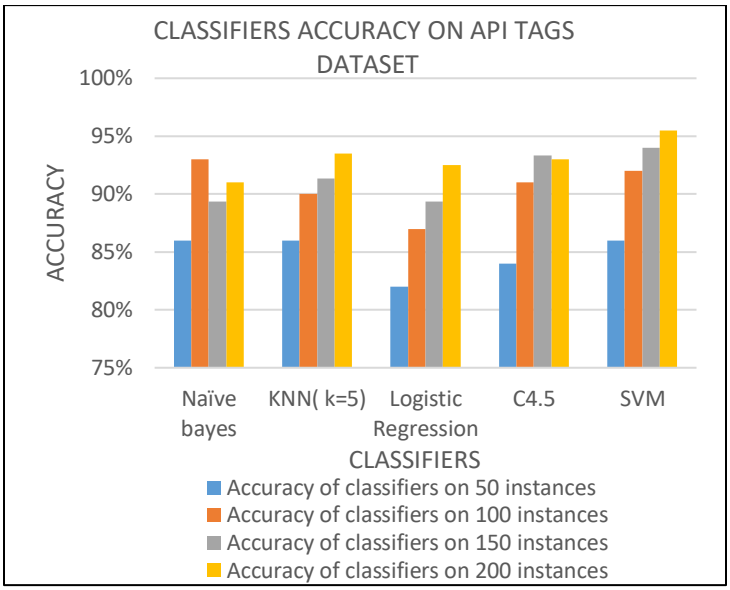

Figure 4: Bar Graph showing classifier wise accuracy for API Tags based analysis

In API Tags based analysis we found that the accuracy of SVM is highest i.e., 95.5% after that Logistic regression and C4.5 give better result in terms of accuracy while KNN holds the lowest accuracy i.e. 86 % in case of 50 instances. Also, we found that in most of the cases ,as we increase the number of instances the accuracy of classifier increases with the slight exception in case of naïve Bayes where the accuracy attain a peak at dataset of 100 instances

Table 3: The result of each classifier on Permission plus API Tags based dataset

Classifier	**Dataset of 50 instances**		**Dataset of 100 instances**		**Dataset of 150 instances**		**Dataset of 200 instances**	
	Acc.	**TPR**	**Acc.**	**TPR**	**Acc.**	**TPR**	**Acc.**	**TPR**
Naïve Bayes	96%	0.960	96%	1.000	94.66 %	0.987	96%	0.990
KNN(k=5)	94%	0.960	96%	0.960	96%	0.960	97.5 %	0.980
Logistic Regressio n	96%	0.960	98%	1.000	96.66 %	0.960	98%	0.980
C4.5	88%	0.920	96%	0.980	95.33 %	0.947	98%	0.990
SVM	96%	1.000	96%	0.940	97.33 %	0.960	98.5 %	0.980

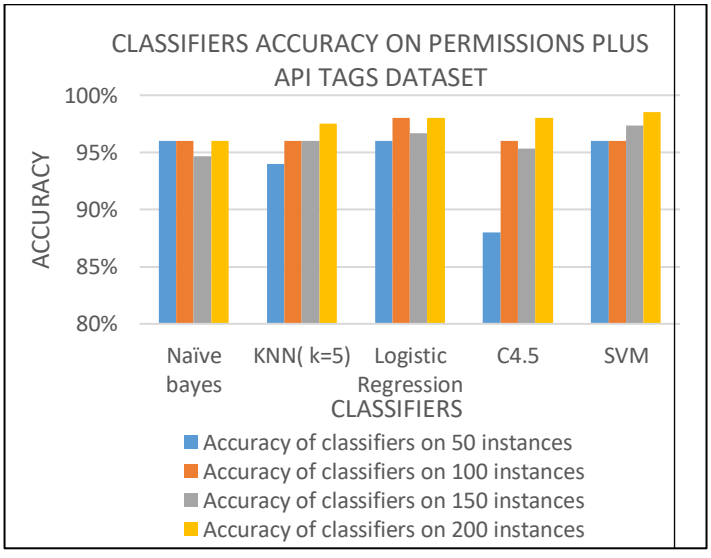

Figure 5: Bar Graph showing classifier wise accuracy for permission plus API Tags based analysis

ISSN: 2233-7857 IJFGCN

In permissions plus API Tags based analysis we found that SVM turns out to be the best classifier as the accuracy of SVM is highest i.e. 98.5. Also, we found that in most of the cases as we increase the number of instances the accuracy of the classifier also increases.

2nd Experiment result:

We found the top ten permissions from each of the database of 100 malicious apps and 100 benign apps collected separately.

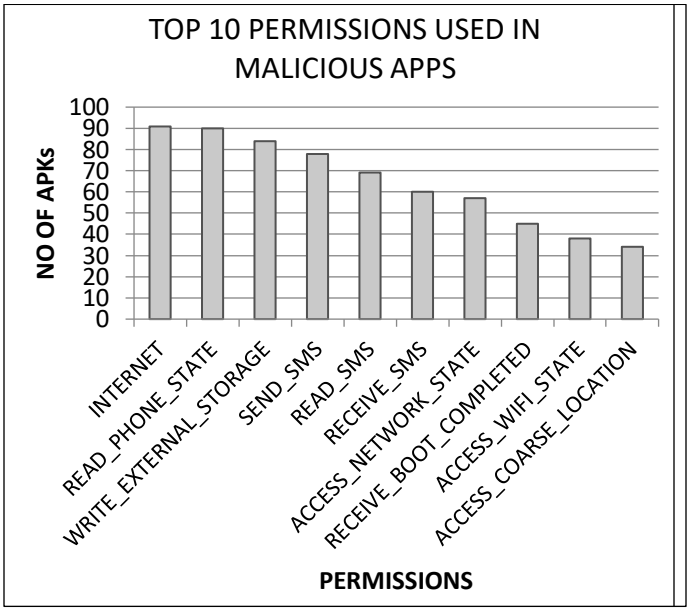

Figure 6: Bar graph showing top 10 permissions used in malicious apps

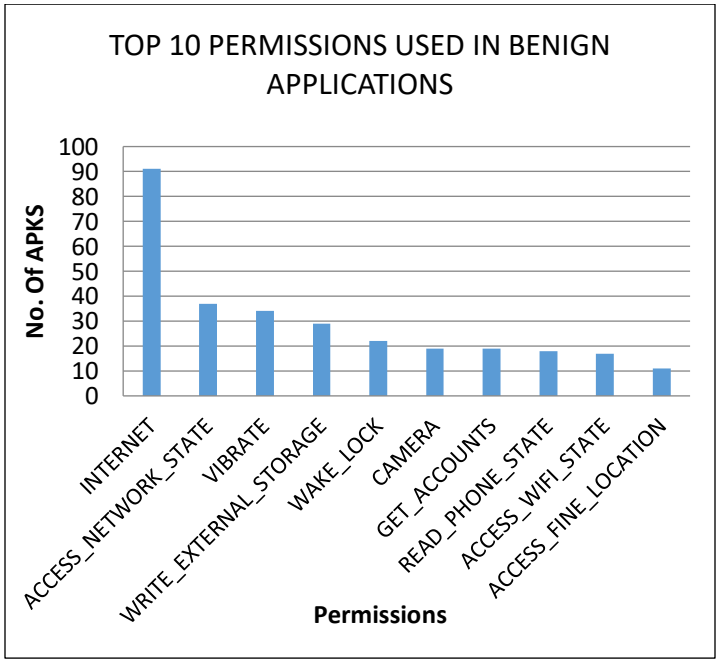

Figure 7: Bar graph showing top 10 permissions used in benign apps

ISSN: 2233-7857 IJFGCN

We also found the top ten API Tags from each of the database of 100 malicious apps and 100 benign apps collected separately.

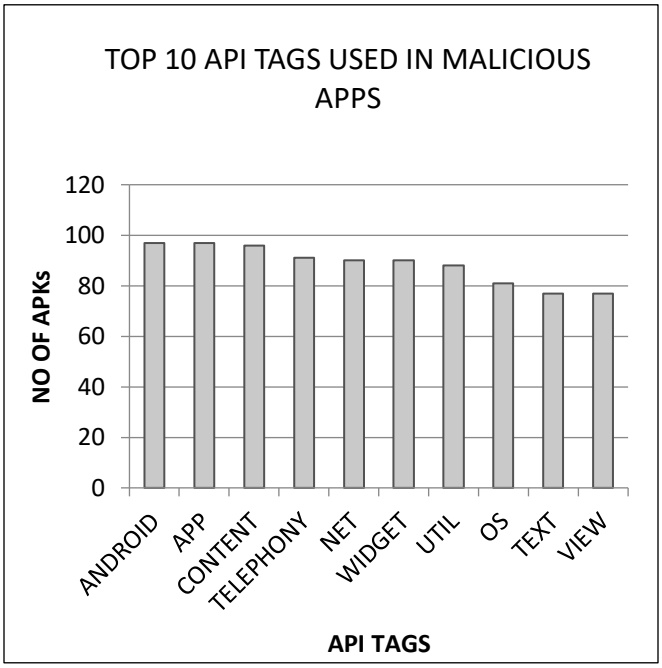

Figure 8: Bar graph showing top 10 API Tags used in malicious apps

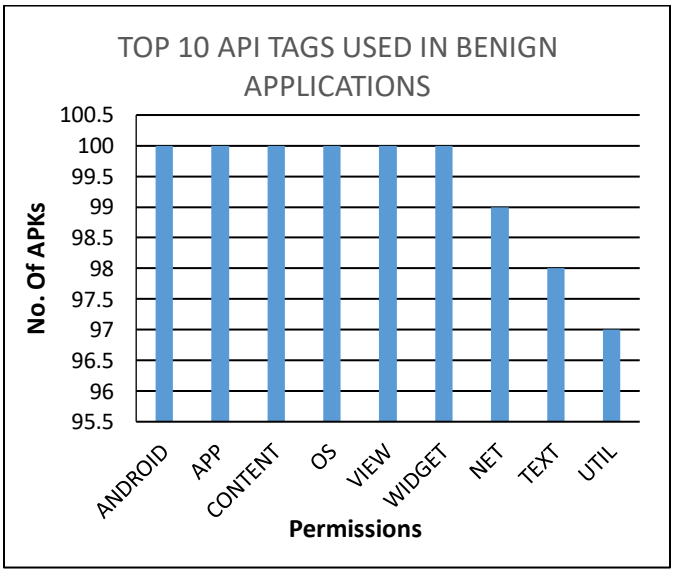

Figure 9: Bar graph showing top 10 API Tags used in benign apps

After comparing figure 6 and figure 7 we conclude that SEND_SMS, READ_SMS, RECEIVE_SMS, RECEIVE_BOOT_COMPLETED and ACCESS_COARSE_LOCATION are the permissions mostly used only in malicious applications. Similarly, from figure 8 and figure 9 we can see that TELEPHONY is the API Tag mostly used only in malicious applications.

5. Conclusion

ISSN: 2233-7857 IJFGCN

In this paper, we have examined about Android applications comprised of both malicious and non-malicious. In Malicious apps, we have used ten malware families which have already been discussed in earlier chapters.

We have used five classification algorithms to detect the malware and subsequently evaluated the classifiers on the basis of accuracy.

Taking into the consideration all databases, SVM classifier turned out to be the best classifier in detecting the malware.

Out of all the three datasets i.e. Permissions, API Tags, Permissions plus API Tags as features, Permission plus API Tags based features comes out be the best for the detection of malware on the basis of features.

Analyzing all the datasets thoroughly, the top ten Android Permissions used in malicious applications are INTERNET, READ_PHONE_STATE, WRITE_EXTERNAL_STORAGE, SEND_SMS, READ_SMS, RECEIVE_SMS, ACCESS_NETWORK_STATE, RECEIVE_BOOT_COMPLETED, ACCESS_WIFI_STATE, ACCESS_COARSE_LOCATION.

Top ten API Tags used in malicious applications are ANDROID, APP, CONTENT, TELEPHONY, NET, WIDGET, UTIL, OS, TEXT, VIEW

After comparing the top most permissions and API Tags in both malicious and benign apps we conclude that SEND_SMS, READ_SMS, RECEIVE_SMS, RECEIVE_BOOT_COMPLETED and ACCESS_COARSE_LOCATION are the dangerous permissions and TELEPHONY is the dangerous API Tag.

References

[1] http://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

[2] http://www.androidcentral.com/google-says-there-are-now-14-billion-active-android-devices-worldwide

[3] http://www.techi.com/2015/12/india-now-has-more-than-a-billion-smartphone-users/

[4]http://www.scmagazineuk.com/updated-97-of-malicious-mobile-malware-targets-android/article/422783/

[5] Odusami, Modupe, Olusola Abayomi-Alli, Sanjay Misra, Olamilekan Shobayo, Robertas Damasevicius, and Rytis Maskeliunas. "Android malware detection: A survey." In *International Conference on Applied Informatics*, pp. 255-266. Springer, Cham, 2018.

[6] Xiao, Xi, Shaofeng Zhang, Francesco Mercaldo, Guangwu Hu, and Arun Kumar Sangaiah. "Android malware detection based on system call sequences and LSTM." *Multimedia Tools and Applications* 78, no. 4 (2019): 3979-3999.

[7] Milosevic, Nikola, Ali Dehghantanha, and Kim-Kwang Raymond Choo. "Machine learning aided Android malware classification." *Computers & Electrical Engineering* 61 (2017): 266-274.

[8] Li, Jin, Lichao Sun, Qiben Yan, Zhiqiang Li, Witawas Srisa-An, and Heng Ye. "Significant permission identification for machine-learning-based android malware detection." *IEEE Transactions on Industrial Informatics* 14, no. 7 (2018): 3216-3225.

[9] Demontis, Ambra, Marco Melis, Battista Biggio, Davide Maiorca, Daniel Arp, Konrad Rieck, Igino Corona, Giorgio Giacinto, and Fabio Roli. "Yes, machine learning can be more secure! a case study on

ISSN: 2233-7857 IJFGCN

android malware detection." *IEEE Transactions on Dependable and Secure Computing* 16, no. 4 (2017): 711-724.

[10] Martín, Ignacio, José Alberto Hernández, and Sergio de los Santos. "Machine-Learning based analysis and classification of Android malware signatures." *Future Generation Computer Systems* 97 (2019): 295-305.

[11] Chen, Xiao, Chaoran Li, Derui Wang, Sheng Wen, Jun Zhang, Surya Nepal, Yang Xiang, and Kui Ren. "Android HIV: A study of repackaging malware for evading machine-learning detection." *IEEE Transactions on Information Forensics and Security* 15 (2019): 987-1001.

[12] Sanz, B., Santos, I., Laorden, C., Ugarte-Pedrero, X., Bringas, P.G., Alvarez, G.,: PUMA: Permission Usage to detect Malware in Android, Springer, Berlin Heidelberg, 2013

[13] Justin Sahs , Latifur Khan, A Machine Learning Approach to Android Malware Detection, Proceedings of the 2012 European Intelligence and Security Informatics Conference, p.141-147, August 22-24, 2012

[14] Zarni Aung, Win Zaw. “Permission-Based Android Malware Detection” in International Journal of Scientific & Technology Research Volume 2, Issue 3, March 2013

[15] Aafer, Y., Du, W., Yin, H.: DroidAPIMiner: Mining API-level features for robust malware detection in android. In: Zia, T., Zomaya, A., Varadharajan, V., Mao, M. (eds.) SecureComm 2013. LNICST, vol. 127, pp. 86–103. Springer, Heidelberg (2013)

[16] https://play.google.com/store?hl=en

[17] Akanksha Sharma and Subrat Kumar Dash,: Mining API Calls and Permissions for Android Malware Detection, proceedings of 13th International Conference, CANS 2014, Heraklion, Crete, Greece, October 22-24, 2014.

[18] http://sanddroid.xjtu.edu.cn:8080/

[19] https://github.com/androguard/androguard

[20] Jung, Jaemin, Hyunjin Kim, Dongjin Shin, Myeonggeon Lee, Hyunjae Lee, Seong-je Cho, and Kyoungwon Suh. "Android malware detection based on useful API calls and machine learning." In *2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pp. 175-178. IEEE, 2018.

[21]Gong, Zhiqiang, Ping Zhong, and Weidong Hu. "Diversity in machine learning." *IEEE Access* 7 (2019): 64323-64350.

[22]Cai, Jie, Jiawei Luo, Shulin Wang, and Sheng Yang. "Feature selection in machine learning: A new perspective." *Neurocomputing* 300 (2018): 70-79.

[23] www.cs.waikato.ac.nz/ml/weka/

[24]A. Dey, ‘‘Machine learning algorithms: A review,’’ Int. J. Comput. Sci. Inf. Technol., vol. 7, no. 3, pp. 1174–1179, 2016.

ISSN: 2233-7857 IJFGCN

[25]M. Mohri, A. Rostamizaden, and A. Talwalkar, Foundations of Machine Learning, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

ISSN: 2233-7857 IJFGCN