Group-Wise Classification Approach to Improve Android Malicious Apps Detection Accuracy

Ashu Sharma and Sanjay Kumar Sahay
(Corresponding author: Sanjay K. Sahay)

Birla Institute of Technology and Science, Pilani, Department of Computer Science and Information Systems
Goa Campus, NH-17B, By Pass Road, Zuarinagar-403726, Goa, India
(Email: ssahay@goa.bits-pilani.ac.in)

(Received Dec. 21, 2017; Revised and Accepted Mar. 7, 2018; First Online Jan. 14, 2019)

Abstract

In the fast-growing smart devices, Android is the most popular OS, and due to its attractive features, mobility, ease of use, these devices hold sensitive information such as personal data, browsing history, shopping history, financial details, *etc.* Therefore, any security gap in these devices means that the information stored or accessing the smart devices are at high risk of being breached by the malware. These malware are continuously growing and are also used for military espionage, disrupting the industry, power grids, *etc.* To detect these malware, traditional signature matching techniques are widely used. However, such strategies are not capable to detect the advanced Android malicious apps because malware developer uses several obfuscation techniques. Hence, researchers are continuously addressing the security issues in the Android based smart devices. Therefore, in this paper using *Drebin* benchmark malware dataset we experimentally demonstrate how to improve the detection accuracy by analyzing the apps after grouping the collected data based on the permissions and achieved 97.15% overall average accuracy. Our results outperform the accuracy obtained without grouping data (79.27%, 2017), Arp, *et al.* (94%, 2014), Annamalai *et al.* (84.29%, 2016), Bahman Rashidi *et al.* (82%, 2017)) and Ali Feizollah, *et al.* (95.5%, 2017). The analysis also shows that among the groups, *Microphone* group detection accuracy is least while *Calendar* group apps are detected with the highest accuracy, and for the best performance, one shall take 80-100 features.

Keywords: Android Malicious Apps; Dangerous Permissions; Machine Learning; Static Malware Analysis

1 Introduction

The attractive features and mobility of smart devices have drastically changed the today's environment. Many functionalities of these devices are similar to the traditional information technology system, which can also access enterprises applications and data, enabling employees to do their work remotely. Hence the security risks are not only limited to Bring Your Own Smart Device (BYOSD) scenarios but also for the devices which are adopted on an ad hoc basis. Therefore, any security gap in these devices means that the information stored or accessing smart devices are at high risk of being breached. The recent attack shows that the security features in these devices are not as par to completely stop the adversary [23]. Hence smart devices are becoming an attractive target for the online criminal, and they are investing more and more for the sophisticated attacks viz. ransomware or to steal the valuable personal data from the user device.

In the smart devices, Android is the most popular operating systems and are connected through the internet accessing billions of online websites (an estimate shows that 5 out of 6 mobile phones are working on Android OS [25]). Its popularity is basically due to its open source, exponential increase in the Android supported apps, third-party distribution, free rich SDK and the very much suited Java language. In this growing Android apps market, it is very hard to know which apps are malicious. As per Statista [24], there are approximately two million apps at the *Play Store* of Google and also many third-party apps available for the Android users. Hence potential of the malicious apps or malware entering these systems is now at never seen before levels, not only to the normal users but also for military espionage, disrupting the industry, power grids (e.g., Duqu, StuxNet), *etc.* [21]. In this, Quick Heal Threat Research Labs in the 3rd quarter of 2015 reported that they had received $\sim 4.2 \times 10^5$ malware per day for the Android and Windows platforms [15].

To detect the malware, traditional approaches are based on the signature matching, which is efficient from a time perspective but not relevant for the detection of advanced malicious apps and continuously growing zero-day malware attack [9]. Also, to evade the signature-based techniques, malware developer uses several obfuscation techniques. However, to detect the Android malicious apps, time to time, a number of static and dynamic methods have been proposed [2,5,11,16]. But, it appears that the proposed methods are not good enough to effectively detect the advanced malware [21] in the fast-growing internet and Android based smart devices usage into our daily life. Hence researchers are continuously addressing the security issues in the Android based smart devices. Therefore, in this paper, for the effective detection of Android malicious apps with high accuracy, we classified the apps after grouping the collected data based on permissions. The remaining paper is organized as follows. In next Section, we discuss the related work. Section 3 describes how the collected Android apps are grouped, Section 4 explains the feature selection approach, while Section 5 describes our approach for the effective detection of Android malicious apps and the obtained experimental results. Finally, Section 6 contains the conclusion.

2 Related Work

In both the two main methods (static and dynamic) used for the classification of malicious apps, selected classifiers are trained with a known dataset to differentiate the benign and malicious apps. In this, Arpil *et al.* achieved 94% detection accuracy by generating a joint vector space using AndroidManifest.xml file and the disassembled code [2]. Seo, *et al.* also used the same static features viz. permissions, dangerous APIs, and keywords associated with malicious behaviors to detect potential malicious apps [19].

Based on a set of characteristics derived from binary and metadata Gonzalez, *et al.* proposed a method *DroidKin*, which can detect the similarity among the apps under various levels of obfuscation [6]. Quentin *et al.*, uses op-code sequences to detect the malicious apps. However, their approaches are not suitable to detect the malware which are completely different [8].

In 2015, Smita Naval, *et al.* proposed an approach by quantifying the information-rich call sequences to detect the malicious binaries and claimed that the model is less vulnerable to call-injection attacks [12]. In 2016, Jaewook jang, *et al.* proposed *Andro-Dumpsys*, a hybrid malware detection approach based on the similarity between the malware creator-centric and malware-centric information. Their experimental analysis shows that *Andro-Dumpsys* can classify the malware families with good True Positive (TP) and True Negative (TN), and are also capable of identifying zero-day threats [7]. Luca Caviglione, *et al.* obtained 95.42% accuracy using neural networks and decision trees [12].

Sanjeev Das, *et al.* proposed *GuardOl* (a hardware-enhanced architecture), a combined approach using processor and field programmable gate array for online malware detection. Their approach detects 46% of malware for the first 30% of execution, while 97% on complete execution [4]. Saracino, *et al.*, proposed a host-based malware detection system called MADAM which simultaneously analyzes and correlates the features at four levels to detect the malware [18]. Gerardo Canfora, *et al.* analyzed two methods to detect Android malware, first was based on Hidden Markov Model, while the 2nd one exploits structural entropy and found that the structural entropy can identify the malware family more correctly [3].

Annamalai *et al.* proposed *DroidOl* for the effective online detection of malware using passive-aggressive classifier and achieved an accuracy of 81.29% [11].

Recently in 2017, Feizollah, *et al.* evaluated the effectiveness of Android Intents (explicit and implicit) as a distinguishing feature for identifying malicious applications. They conducted experiments using a dataset containing 7406 applications comprising 1846 clean and 5560 infected applications. They achieved the detection rate of 91% using Android Intent and 83% using Android permission. With the combination of both the features, they have achieved 95.5% detection rate [5]. Nikola *et al.* estimated F-measure (*does not take account of correctly classified benign apps*) of 95.1% and 89% by classifying the apps based on source code and permission respectively [10].

Rashidi *et al.* experimented with the *Drebin* benchmark malware dataset and shown that their model can accurately assess the risk levels of malicious applications and provide adaptive risk assessment based on user input and can find malware with the maximum accuracy of 82% [16].

3 Grouping of Android Apps

In Android, apps run as a separate process with unique user/group ID and operate in an application sandbox so that apps execution can be kept in isolation from other apps and the system. Hence, to access the user data/resources from the system, apps need additional capabilities that are not provided by the basic sandbox. To access data/resources which are outside of the sandbox, the apps have to explicitly request the needed permission. Depending on the sensitivity of data/area, requested permission may be granted automatically by the system or ask the user to approve or reject the request. In Android, these permissions can be found in Manifest.permission file e.g. to use the call service in an Android app, it should specify:

```
< manifestxmlns :
Android = "http : //schemas.Android.com/apk/res/Android"
package = "com.Android.app.callApp" >
< uses − permissionAndroid :
name = "android.permission.CALL_PHONE" / >
...
< /manifest >
```

In total there are 235 permissions out of which 163 are hardware accessible and remaining are for user information access [13]. In terms of security, all these permissions can be put into two categories i.e. normal and dangerous permissions [1]. Therefore it will be important to study the classification of Android malicious apps after grouping them into dangerous and normal/other permissions (Table 1). Hence in this paper to improve the overall average detection accuracy of Android malicious apps we use *Drebin* [2] 5531 benchmark malware dataset and 4235 benign apps available at Google play store. Our analysis shows that the *Drebin* dataset does not contain any apps which need body sensors permission.

Therefore we ignored the Sensors group in our experimental analysis and made total nine groups (eight groups of dangerous permissions and one group of normal/other permissions) for the detection of Android apps.

Table 1: Dangerous permissions groups of the Android apps

Group	Permissions
Calendar	Read calendar and write calendar.
Camera	Use camera.
Contacts	Read contacts, write contacts and get contacts.
Location	Access fine location and Access coarse location.
Microphone	Record audio.
Phone	Read phone state, call phone, read call logs, add voicemail, use sip and process outgoing calls.
Sensors	Use body sensors
SMS	Send SMS, receive SMS, read SMS receive WAP push and receive MMS.
Storage	Read external storage and write external storage.

4 Feature Selection

For the detection of Android malicious apps, feature selection plays a vital role, not only to represent the target concept but also to speed-up the learning and testing process. In this, often datasets are represented by many features. However, few of them may suffice to improve the concept quality, and also limiting the features will speed-up the classification. The Android apps can be represented as a vector of 256 opcodes [14], and some of these opcodes can be used as features for the effective and efficient detection of Android malicious apps. Therefore, to find the prominent features which can represent the target concept, opcodes from the collected Android apps are extracted as follows

- The *.apk* files (Android apps) has been decompiled by using freely available *apktool*;
- From the decompiled data, we kept only the *.smali* files and discarded other data, and then;
- Opcodes are extracted from the *.smali* files.

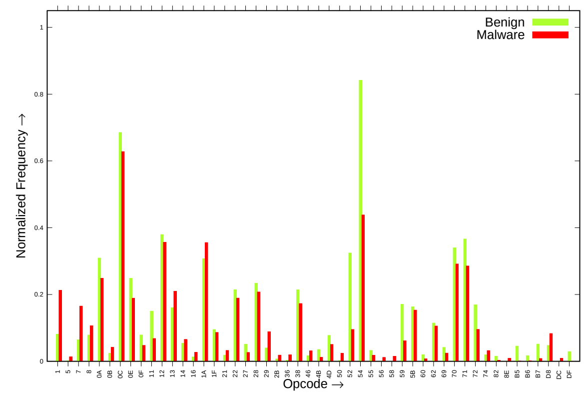

Figure 1: Top 50 opcodes occurrence difference between benign and malicious apps in the Calendar group

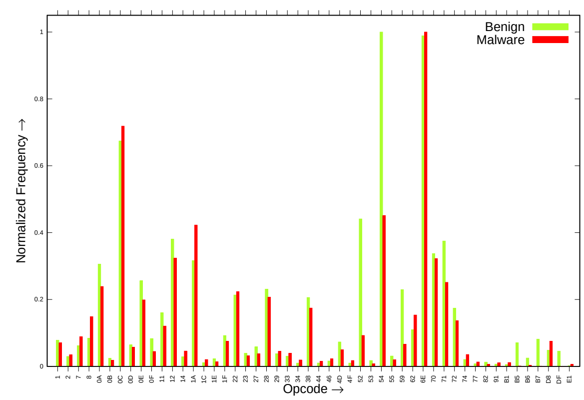

Figure 2: Top 50 opcodes occurrence difference between benign and malicious apps in the Camera group

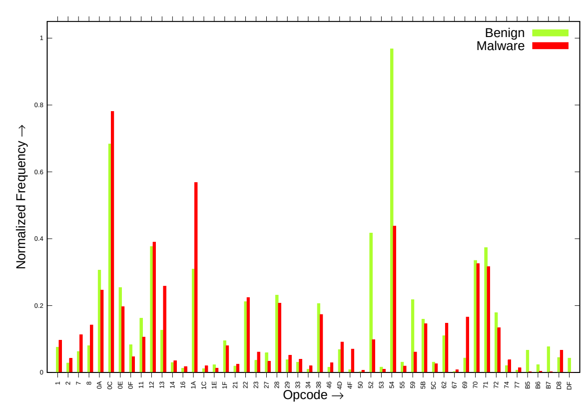

Figure 3: Top 50 opcodes occurrence difference between benign and malicious apps in the Contacts group

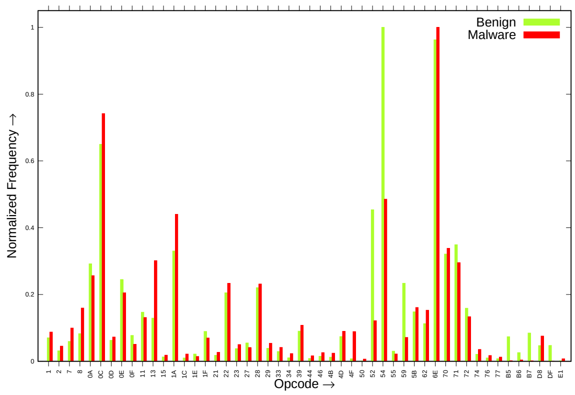

Figure 4: Top 50 opcodes occurrence difference between benign and malicious apps in the Location group

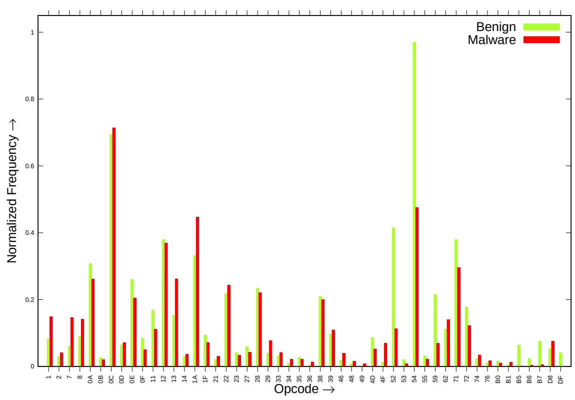

Figure 5: Top 50 opcodes occurrence difference between benign and malicious apps in the Microphone group

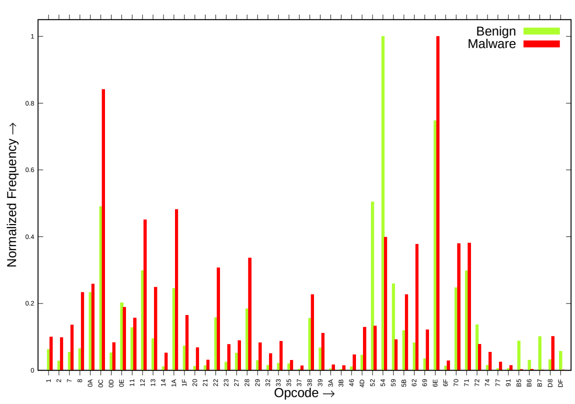

Figure 6: Top 50 opcodes occurrence difference between benign and malicious apps in the Other group

We studied the occurrence of opcodes in benign and malicious apps separately in each formed group, and computed the opcode occurrences difference between them. We observe that the opcode occurrence between malicious and benign apps among the formed group differ significantly (group-wise top 50 opcodes whose occurrence significantly differ are shown in Figures 1 - 9 for the *Calendar, Camera, Contacts, Location, Microphone, Others, Phone, SMS,* and *Storage* group respectively). Also, we find that the opcode occurrence in any group differs significantly when compared with the opcode occurrence obtained without forming the groups [22].

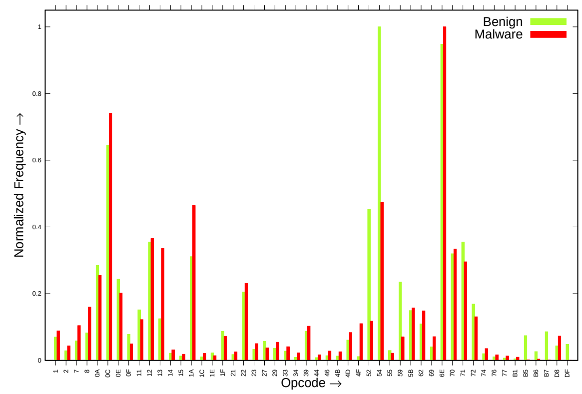

Figure 7: Top 50 opcodes occurrence difference between benign and malicious apps in the Phone group

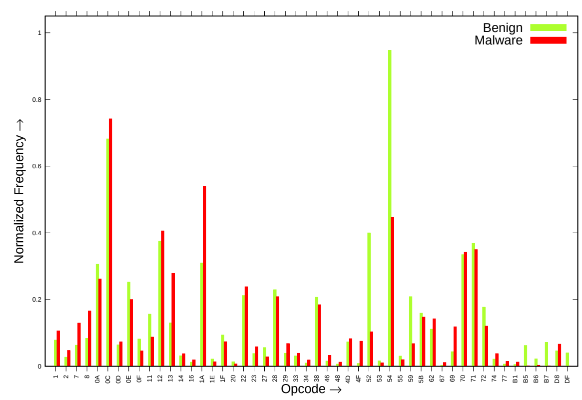

Figure 8: Top 50 opcodes occurrence difference between benign and malicious apps in the SMS group

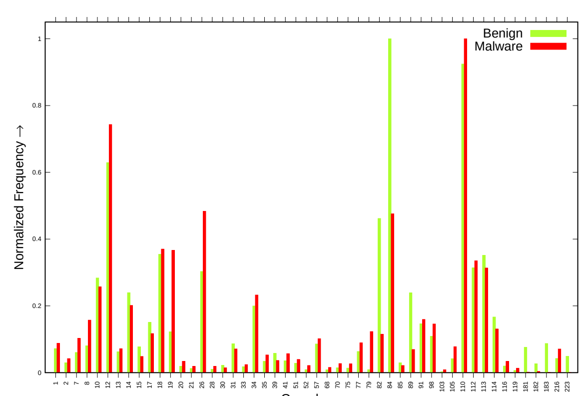

Figure 9: Top 50 opcodes occurrence difference between benign and malicious apps in the Storage group

Hence, the final features are selected after ordering the opcodes by their occurrence difference in each group (Al-

Table 2: Number of benign and Android malicious apps used for training and testing the classifiers

Groups	Train malware	Train benign	Test malware	Test benign	Total No. of apps
Calendar	59	57	14	14	144
Camera	179	423	44	106	752
Contacts	1073	356	268	89	1786
Location	1538	68	383	18	2007
Microphone	95	218	23	55	391
Others	110	891	27	223	1251
Phone	3981	1453	986	373	6793
SMS	2712	239	677	60	3688
Storage	2923	837	730	210	4700

gorithm 1) and used it for the detection of Android malicious apps.

Algorithm 1 : Feature Selection

INPUT: Pre-processed data
$\mathbf{N_B}$: No. of benign apps, $\mathbf{N_M}$: No. of malicious apps,
n: Total number of features required.
OUTPUT: List of features

BEGIN
for all benign and malicious apps **do**
Find the sum of frequencies $\mathbf{f_i}$ of each opcode **Op** and normalize it.

$$F_B(Op_j) = (\sum f_i(Op_j))/N_B$$

$$F_M(Op_j) = (\sum f_i(Op_j))/N_M$$

end for
for all opcode $\mathbf{Op_j}$ **do**

$$D(Op_j) = |F_B(Op_j) - F_M(Op_j)|$$

end for
return **n** number of prominent opcodes as features with high $\mathbf{D(Op)}$.

5 Classification of Malicious Apps

Ashu *et al.* [22] without grouping the data nor talking the apps permission investigated the top five classifiers viz. FT, RF, LMT, NBT and J48 for the classification of apps and reported that the FT is the best classifier and can detect the malicious apps with 79.27% accuracy [22]. Hence to improve the detection accuracy in this paper, first we grouped the apps based on the permissions and then classify the malicious apps using prominent opcode as the features (Figure 10). For the classification, the detail distribution (No. of training and testing malicious/benign apps, No. of apps in the group used for the classification) of the total collected dataset is given in Table 2. For the group-wise classification, we have used Waikato Environment for Knowledge Analysis (WEKA).

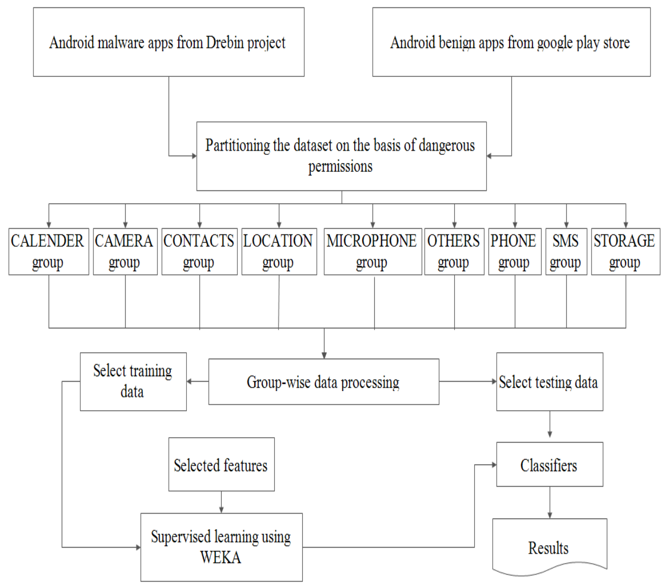

Figure 10: Flow chart for the detection of Android malicious apps by grouping the data

On the basis of studies [17, 20], we selected the same classifier (FT, RF, LMT, NBT, and J48) for the classification, but prominent features, training, and testing data are taken from the formed group only (Table 2). To measure the goodness of trained models, we evaluate the detection accuracy given by the equation

$$\text{Accuracy}(\%) = \frac{\text{True Positive} + \text{True Negative}}{\text{Total No. of Android Apps}} \times 100.$$

Where True Positive/Negative is the Android malicious/benign apps correctly classified [22].

The performance of the classifier has been investigated for each group by taking randomly 20% of the collected data (other than the training) with 20 - 200 best features incrementing 20 features at each step and the result obtained are shown in Figures 11 - 19 for the *Calendar, Camera, Contacts, Location, Microphone, Others, Phone, SMS,* and *Storage* group respectively.

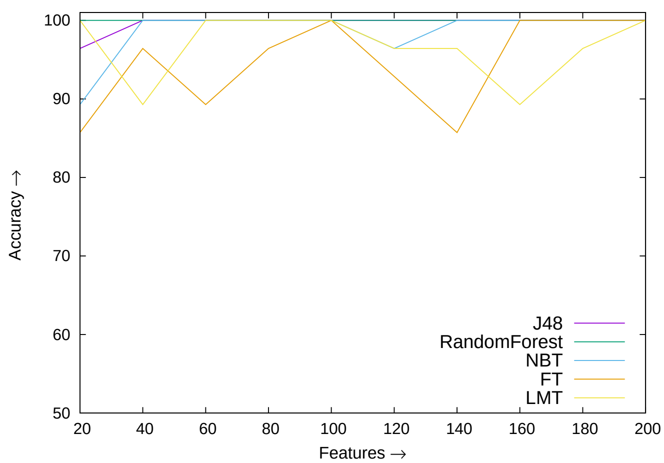

Figure 11: Detection accuracy obtained by the selected five classifiers for the Calendar group

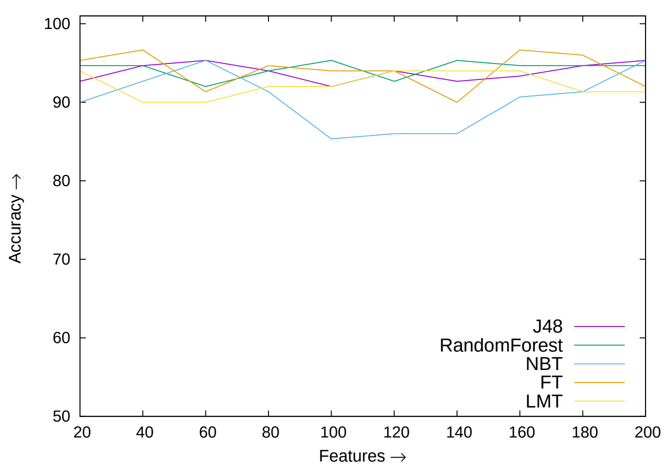

Figure 12: Detection accuracy obtained by the selected five classifiers for the Camera group

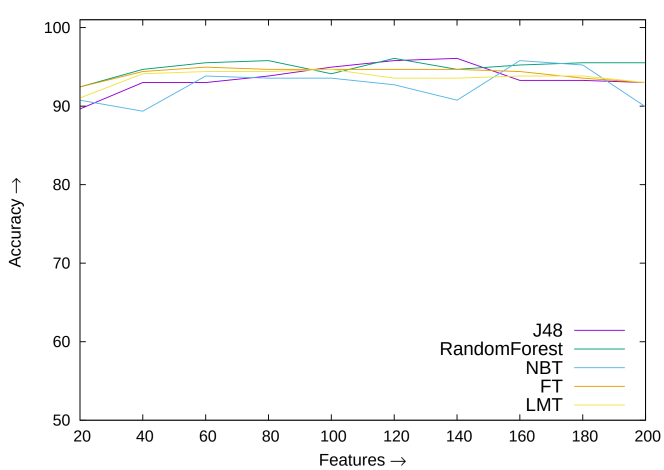

Figure 13: Detection accuracy obtained by the selected five classifiers for the Contacts group

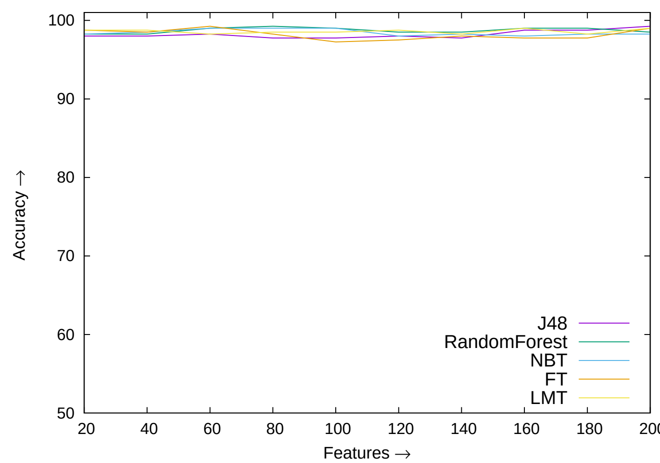

Figure 14: Detection accuracy obtained by the selected five classifiers for the Location group

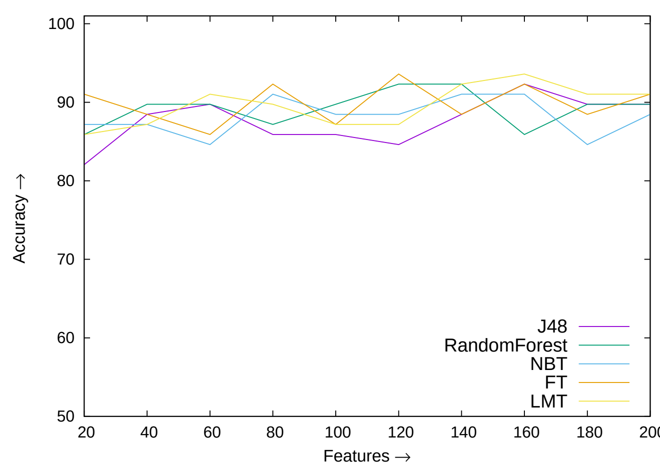

Figure 15: Detection accuracy obtained by the selected five classifiers for the Microphone group

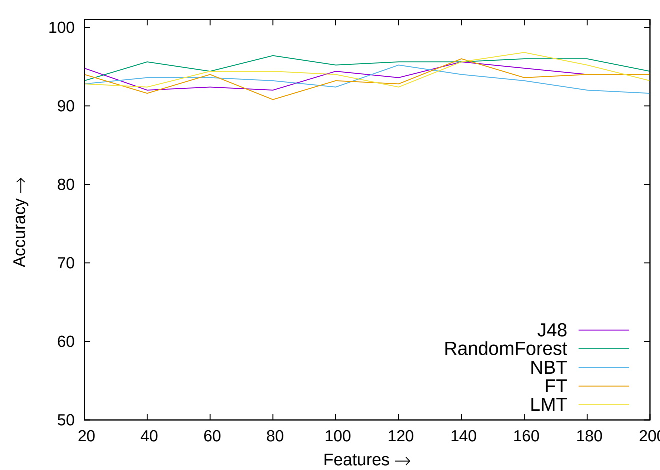

Figure 16: Detection accuracy obtained by the selected five classifiers for the Others group

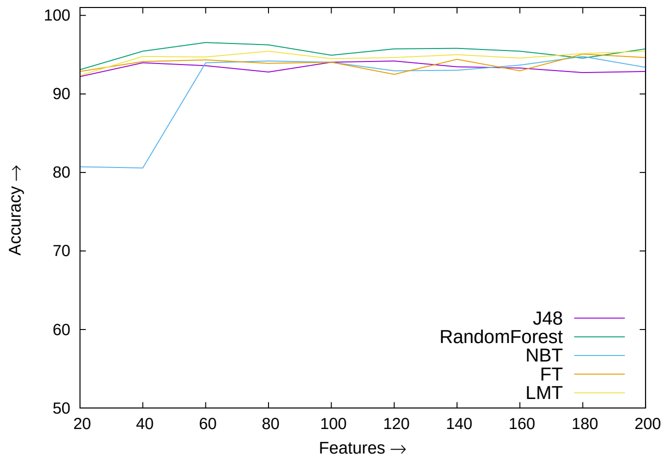

Figure 17: Detection accuracy obtained by the selected five classifiers for the Phone group

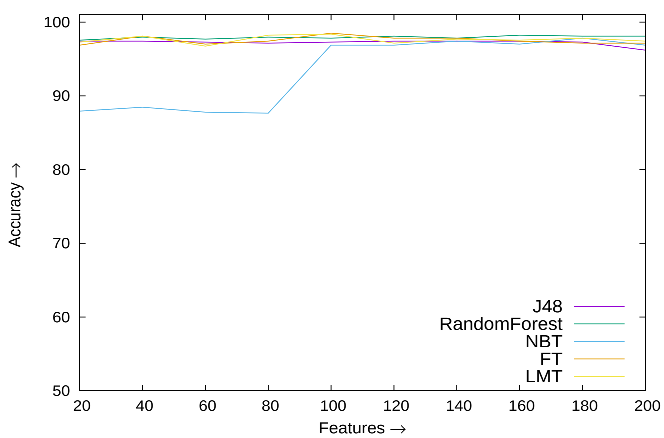

Figure 18: Detection accuracy obtained by the selected five classifiers for the SMS group

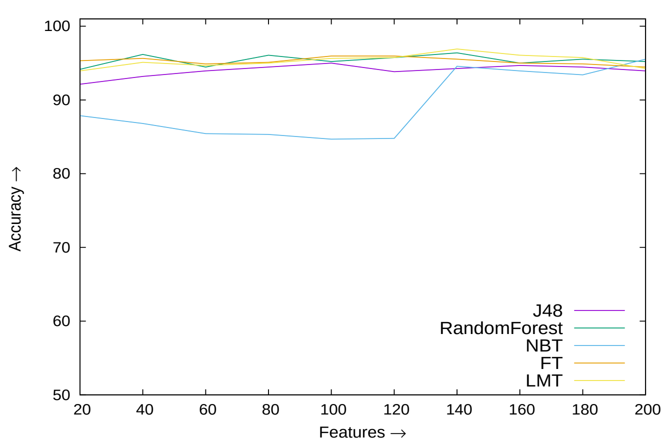

Figure 19: Detection accuracy obtained by the selected five classifiers for the Storage group

Table 3: Average accuracy obtained by the five classifiers

No. of Features	J48	RF	NBT	FT	LMT
20	93.69	95.01	90.37	93.32	94.28
40	95.28	96.26	92.26	93.78	93.45
60	95.51	96.10	94.24	94.01	94.31
80	94.83	**96.32**	94.44	95.38	95.46
100	**95.15**	96.24	94.41	**95.43**	**85.47**
120	94.48	95.96	92.96	94.57	94.23
140	95.12	96.08	93.68	93.53	94.76
160	95.39	95.16	**94.97**	95.16	94.29
180	94.94	95.73	93.93	95.18	94.56
200	94.71	95.78	93.24	94.98	94.71
Maximum	**95.51**	**96.32**	**94.97**	**95.43**	**95.47**
Minimum	93.69	95.01	90.37	93.32	93.45

The average accuracy obtained by the selected classifier are shown in Table 3. Here, the average accuracy means the sum of accuracy obtained by the classifier in the individual group with a fixed number of features divided by the total number of groups.

The analysis shows that RF average detection accuracy is best among the five classifiers and fluctuates least with the number of features, whereas NBT performance is worst and fluctuate maximum with the number of features.

However, the maximum average accuracy obtained by the selected five classifiers does not fluctuate much (94.97% - 96.32%) but minimum average accuracy fluctuation is high (90.37% - 95.01%), and for the best performance one shall take top 80 - 100 features, for the training and testing. The best accuracy obtained by the classifier in all the groups are given in Table 4.

We find that the detection accuracy is maximum in the Calendar group and minimum in the Microphone group obtained by FT and RF classifier respectively. The overall average maximum accuracy comes to 97.15%, which is very much better than then the obtained accuracy without grouping and taking permissions into account [22] and Arp, *et al.* (94%, 2014), Annamalai *et al.* (84.29%, 2016), Bahman Rashidi *et al.* (82%, 2017), Ali Feizollah, *et al.* (95.5%, 2017) (Figure 20).

In terms of TP i.e. detection rate of malicious apps, the *Calendar* group are best classified by RF and *SMS* group are least by FT, while in terms of TN i.e. benign detection rate, *Calendar*, and *SMS* group are best classified with RF and FT classifier respectively, while Others group containing normal permissions is best classified by the LMT classifier. The group-wise results of TP and TN obtained by the classifiers which give the best accuracy are shown in Table 4.

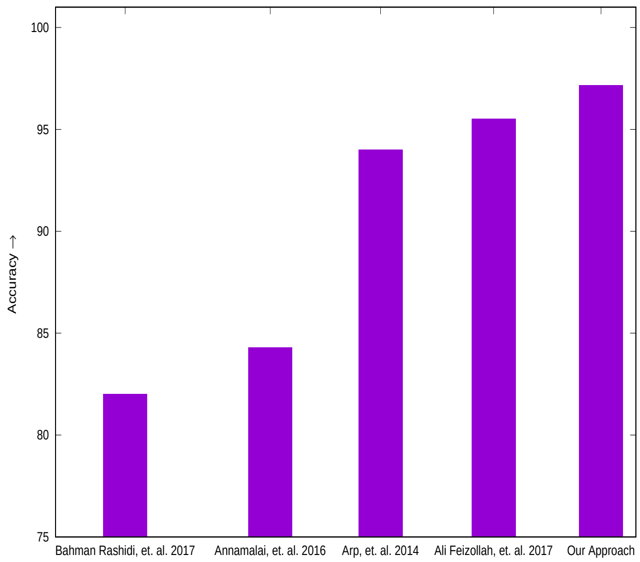

Figure 20: Comparisons of accuracy achieved by us with four other authors

Table 4: Group-wise maximum accuracy, TP and TN obtained by the classifiers

Groups	Best Classifier	Accuracy	Features Required	TN	TP
Calendar	RF	100.00	20	1.00	1.00
Camera	FT	96.67	40	0.93	0.98
Contacts	RF	96.08	120	0.99	0.89
Location	FT	99.25	60	0.99	0.94
Microphone	FT	93.59	120	0.87	0.96
Others	LMT	96.80	160	0.85	0.98
Phone	RF	96.54	60	0.98	0.92
SMS	FT	98.51	100	1.00	0.80
Storage	LMT	96.91	140	0.99	0.88

6 Conclusion

For the smart devices users, millions of Android apps are available at Google Play store and by the third party. Some of these available apps may be malicious. To defend the threat/attack from these malicious apps, a timely counter-measures has to be developed. Therefore, in this paper using *Drebin* benchmark malware dataset we group-wise analyzed the collected data based on permissions and experimentally demonstrated how to improve the detection accuracy of Android malicious apps and achieved 97.15% average accuracy. The obtained results outperformed the accuracy achieved by without grouping the data (79.27%, 2016), Arp, *et al.* (94%, 2014), Annamalai *et al.* (84.29%, 2016), Bahman Rashidi *et al.* (82%, 2017)) and Ali Feizollah, *et al.* (95.5%, 2017). The outperformance of our approach with the compared author results is basically due to the use of logic of the apps resides in the *.smalli* file and developing nine different models for the classification. Among the groups, the *Microphone* group detection accuracy is least while *Calendar* group apps are detected with maximum accuracy and for the best performance, one shall take top 80 - 100 features. In term of TP i.e. detection rate of malicious apps, *Calendar* group is best classified by RF, and *SMS* group is least by FT, while in terms TN i.e. benign detection rate, *Calendar*, and *SMS* group are best classified by RF and FT classifier respectively, while Others group containing normal permissions is best classified by the LMT classifier. It appears that group-wise detection of Android malicious apps will be efficient than without grouping the data. Hence, for the efficient classification of apps, in-depth study is required to optimize the feature selection, identifying the best-suited classifier for the group-by-group analysis. In this direction, work is in progress and will be reported elsewhere.

Acknowledgements

Mr. Ashu Sharma is thankful to BITS, Pilani, K.K. Birla Goa Campus for the Ph.D. scholarship No. Ph603226/July 2012/01. We are also thankful to Technische Universitat Braunschweig for providing the Drebin dataset for research on Android malware.

References

[1] Android-developers, *Normal and Dangerous Permissions requesting permissions*, Technical Report, Android Labs, 2017.

[2] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck, "Drebin: Effective and explainable detection of android malware in your pocket," in *NDSS*, pp. 1–15, 2014.

[3] G. Canfora, F. Mercaldo, and C. A. Visaggio, "An HMM and structural entropy based detector for android malware: An empirical study," *Computers & Security*, vol. 61, pp. 1–18, 2016.

[4] S. Das, Y. Liu, W. Zhang, and M. Chandramohan, "Semantics-based online malware detection: Towards efficient real-time protection against malware," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 2, pp. 289–302, 2016.

[5] A. Feizollah, N. B. Anuar, R. Salleh, S. T. Guillermo, and S. Furnell, "Androdialysis: Analysis of android intent effectiveness in malware detection," *Computers & Security*, vol. 65, pp. 121–134, 2017.

[6] H. Gonzalez, N. Stakhanova, and A. A. Ghorbani, "Droidkin: Lightweight detection of android apps similarity," in *International Conference on Security and Privacy in Communication Systems*, pp. 436–453, 2014.

[7] J. Jang, H. Kang, J. Woo, A. Mohaisen, and H. K. Kim, "Andro-dumpsys: Anti-malware system based on the similarity of malware creator and malware centric information," *computers & security*, vol. 58, pp. 125–138, 2016.

[8] Q. Jerome, K. Allix, R. State, and T. Engel, "Using opcode-sequences to detect malicious android applications," in *2014 IEEE International Conference on Communications (ICC'14)*, pp. 914–919, 2014.

[9] McAfee, *Mcafee Labs Threats Report*, Dec. 2016. (https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-dec-2016.pdf)

[10] N. Milosevic, A. Dehghantanha, and K. K. R. Choo, "Machine learning aided android malware classification," *Computers & Electrical Engineering*, 2017.

[11] A. Narayanan, L. Yang, L. Chen, and L. Jinliang, "Adaptive and scalable android malware detection through online learning," in *International Joint Conference on Neural Networks (IJCNN'16)*, pp. 2484–2491, 2016.

[12] S. Naval, V. Laxmi, M. Rajarajan, M. S. Gaur, and M. Conti, "Employing program semantics for malware detection," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2591–2604, 2015.

[13] K. Olmstead and M. Atkinson, *Apps Permissions in the Google Play Store*, Pew Research Center, 2015. (http://www.pewinternet.org/2015/11/10/apps-permissions-in-the-google-play-store/)

[14] G. Paller, *Dalvik Opcodes*, Android labs, 2017. (http://pallergabor.uw.hu/androidblog/dalvik_opcodes.html)

[15] QuickHeal, *Threat Report 3rd Quarter*, Technical Report, Quick Heal, 2015.

[16] B. Rashidi, C. Fung, and E. Bertino, "Android resource usage risk assessment using hidden markov model and online learning," *Computers & Security*, vol. 65, pp. 90–107, 2017.

[17] S. K. Sahay and A. Sharma, "Grouping the executables to detect malwares with high accuracy," *Procedia Computer Science*, vol. 78, pp. 667–674, 2016.

[18] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli, "Madam: Effective and efficient behavior-based android malware detection and prevention," *IEEE Transactions on Dependable and Secure Computing*, vol. 99, pp. 1–14, 2016.

[19] S. H. Seo, A. Gupta, A. M. Sallam, E. Bertino, and K. Yim, "Detecting mobile malware threats to homeland security through static analysis," *Journal of Network and Computer Applications*, vol. 38, pp. 43–53, 2014.

[20] A. Sharma, S. K. Sahay, and A. Kumar, "Improving the detection accuracy of unknown malware by partitioning the executables in groups," in *Advanced Computing and Communication Technologies*, pp. 421–431, 2016.

[21] A. Sharma and S. K. Sahay, "Evolution and detection of polymorphic and metamorphic malwares: A survey," *International Journal of Computer Applications*, vol. 90, pp. 7–11, Mar. 2014.

[22] A. Sharma and S. K. Sahay, "An investigation of the classifiers to detect android malicious apps," in *Information and Communication Technology*, pp. 207–217, 2017.

[23] A. Shaun, A. Tareq, C. Peter, C. Mayee, and D. Jon, *Internet Security Threat Report 2017*, Symentec, 2017. (https://www.symantec.com/content/dam/symantec/docs/reports/istr-22-2017-en.pdf)

[24] Statista, *Number of Available Applications in the Google Play Store from December 2009 to February 2016*, Technical Report, Statista, Aug. 2016.

[25] Symentec, *Internet Security Threat Report 2016*, Symantec Corporation, 2016. (http://or.himsschapter.org/sites/himsschapter/files/ChapterContent/or/FinnD_ORHIMSS_Spring16_Conf.pdf)

Biography

Mr. Ashu Sharma was born in Jhansi, Uttar Pradesh, India. He received his Bachelor's degree in Computer Science and Engineering from Uttar Pradesh Technical University and Master's degree in Information Security from Atal Bihari Vajpayee Indian Institute of Information Technology and Management, Gwalior. In 2012 he joined the Department of Computer Science and Information Systems, BITS, Pilani, K.K. Birla Goa Campus, India as a full-time research scholar for the Ph.D. degree under the supervision of Dr. Sanjay K. Sahay. He has published several papers in reputed journals and conferences.

Dr. Sanjay Kumar Sahay is working as an Assoicate Professor in the Department of Computer Science and Information Systems, BITS, Pilani, K.K. Birla Goa Campus. He is also a Visiting Associate of IUCAA, Pune. His research interests are Information Security, Data Science, and Gravitational Waves. He basically teaches Network Security, Cryptography, Computer Networks, and Data Mining courses. Before joining BITS, Pilani, and after submitting his Ph.D. thesis on "Studies in Gravitational Wave Data Analysis" during 2002-2003, he continued his work on Data Analysis of Gravitational Waves as a Project Scientist at IUCAA, Pune, India. In 2003-2005 at Raman Research Institute, Bangalore, India he worked as Project Associate on the multi-wavelength astronomy project (ASTROSAT), where he worked on the data pipeline of Scanning Sky Monitor. In 2005 he worked as Post Doctoral Fellow at Tel Aviv University.