Droid Permission Miner: Mining Prominent Permissions for Android Malware Analysis

Aswini A. M.
Department of Computer Science & Engineering
SCMS School of Engineering & Technology
Karukutty, Ernakulam, India
aswinimohan95@gmail.com

Vinod P.
Department of Computer Science & Engineering
SCMS School of Engineering & Technology
Karukutty, Ernakulam, India
pvinod21@gmail.com

Abstract - **In this paper, we propose static analysis of android malware files by mining prominent permissions. The proposed technique is implemented by extracting permissions from 436 .apk files. Feature pruning is carried out to investigate the impact of feature length on accuracy. The prominent features that give way to lesser misclassification are determined using *Bi-Normal Separation (BNS)* and *Mutual Information (MI)* feature selection techniques. Results suggest that Droid permission miner can be used for preliminary classification of Android package files.**

***Keywords* - Androguard, Android malware, Feature extraction, Static analysis**

I. INTRODUCTION

Android is the most popular operating system that has held its root stronger than any other OS in the global smart phone market. The genuine Android applications are available at the Android market whereas third party applications are also developed progressively each day. Due to its ease of modifiability and open source nature, there is a greater chance for the malicious code to be injected to the Android apps. Also, the third party applications have led to the proliferation of the malicious software.

Android devices are vulnerable to threats and some of the noticeable vulnerabilities are denial of service attacks, execution of code by the attackers using Android debugger bridge (adb), stack based buffer overflow resulting in arbitrary code execution, memory corruption to gain root privileges, SQL injection to retrieve useful information, cross site scripting to redirect to other vulnerable domains and to steal user credentials etc. Some other issues are intercepting SMS to gain user credentials and in some cases API's can be queried for retrieving information.

In this paper, we apply static analysis with the aim to understand the significance of permissions for the identification of unseen samples. Investigation is carried out to identify the most prominent permissions that contribute in mobile malware classification with few features.

Remaining sections are organized as follows: Section II provides some related works on static analysis of Android malware; Section III presents the components of Android package; Section IV gives an overview of our proposed methodology, Droid Permission miner; Section V includes the experiments with results and finally the conclusion is discussed in section VI.

II. RELATED WORK

In [13] the authors proposed a hybrid model for identifying maliciousness of an application named ForkBomb using an Android application sandbox. The results showed that ForkBomb used the function `Runtime.Exec()` to execute an external program. The author in [18] proposed a static mechanism for identifying the malicious code in *iCalendar* application. Experiment was performed using ApkTool, WinZip, Dex2Jar and JD-GUI. In [20], a static feature extraction method DroidMat was proposed. The experiment was carried out on malware collected from Contagio mobile [2] and benign apps collected from Google Play. The features like permissions, activities, services, etc inside the manifest file were extracted. Singular value decomposition was applied to determine critical features. An accuracy of 97.87% was achieved in this work.

The authors of [17] proposed permission based static mechanism for detecting the malicious applications. The experiment was performed on 46 pieces of iOS, Symbian 9.x and Android malware. Most common malware activities were found to be selling user information (61%) as well as sending SMS (52%). They also surveyed the availability of root exploits (available for more than 75% of time).

In [16] a kernel based behaviour analysis for Android malware was proposed. It implements a log collector and log analyzer to filter system calls and process trees respectively.

Authors in [14], [15], [21] devised a supervised anomaly detection called Andromaly to extract 88 features. Detection rates were better in case of usage of benign games than benign tools. The authors of [22] proposed a real-time multilevel anomaly detector that monitors Android at kernel level and user level known as MADAM. It used 13 features and implemented machine learning techniques to detect unknown malware. The results depicted that some Trojans, bootkit, Rootkit, spyware etc. were detected at an accuracy of 100%.

In [23] authors proposed DroidAPIMiner to extract API calls using a modified Androguard tool and different classifiers are evaluated using the feature set. They achieved an accuracy of 99% and false positive rate of 2.2% using k-NN classifier.

978-1-4799-2259-14/$31.00©2014

III. ANDROID PACKAGE

The Android architecture [19] is composed of three layers; the operating system (Linux kernel), middleware (open source libraries, run-time environment) and the application framework (Android NDK, SDK, pre-installed core applications). An android package file (.apk) is a zipped file that contains resources and executables. Its components [3] are the following:

A. Android manifest file: It is the XML file that contains information about the permissions provided by the application, metadata, hardware and software features required by application, version, package name etc.

B. Res folder: It contains the resources of application that defines languages, sound settings, graphics layout, attributes, drawables etc.

C. Classes.dex: These are Dalvik virtual machine executables that can be run in Dalvik machine. It includes the compiled Java source code in `.dex` format.

D. Lib: It consists of native libraries (for C or C++) that can be used through NDK (native development kit).

E. META-INF: It includes application's signature mainly aimed at its security and integrity. It comprises of application's certificate, list of resources and manifest file.

F. Resources.arsc: It is obtained as a result of compilation process.

IV. PROPOSED METHODOLOGY

In this section, we discuss our model for static feature extraction mechanism known as Droid Permission Miner that uses *Androguard* [1] for mining permissions for the detection of malicious applications. The features are extracted from the *Android Manifest file* of these samples. Our proposed system (refer Figure 1) considers the most prominent permissions that contribute to the detection of malicious apps. We briefly discuss about this framework in the following subsections.

A. Datasets

A total of 209 malicious samples are collected from the Contagiodump [2]. Likewise, benign applications were downloaded from various Internet sources (227 numbers).

B. Permission Extraction

In the feature extraction phase, the .apk files are given as input to Androguard to generate human readable manifest file which is originally in binary format. The python script `androaxml.py` is used for generating the xml file from the .apk file. In the remaining part of this article, we refer features as permissions.

C. Divide the dataset into test and training set

The complete dataset is divided into train and test set. Out of the total 227 benign samples, 114 samples are included in the training set and 113 samples are included in the test set. From 209 malware samples, 105 samples are included in the training set and the remaining 104 samples are reserved as test set.

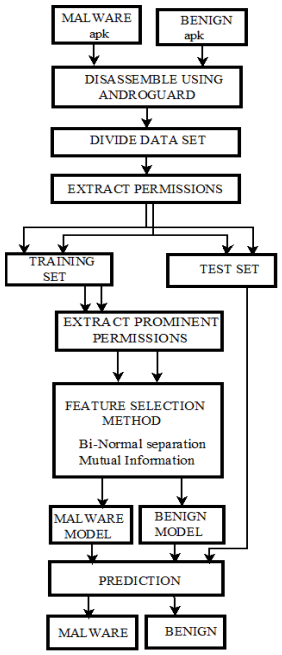

Fig.1. Architecture of Droid Permission Miner

D. Feature categorization

During data preprocessing phase, from the training samples we determine all those permissions that are used by both benign as well as malware applications and filter out features that have minimum probability in identifying the target classes. From these synthesized permissions, we create four categories of permission sets:

1. Malware and benign features $(M \cup B)$
2. Common to malware and benign features $(M \cap B)$
3. Discriminant malware features $(M \setminus B)$
4. Discriminant benign features $(B \setminus M)$

E. Feature selection using BNS and MI

Feature selection is performed to extract a subset of k best features from a set of large feature space consisting of n features. Dimensionality reduction is usually employed so as to eliminate attributes/variables (appearing as noise) which do not contribute in identifying target class. We have used *Bi-Normal separation* [8], [9] (is based on *Z-Score* [4]) and *Mutual Information (MI)* [5] as the feature selection methods. BNS is a Normal cumulative distribution function designated as F and F^{-1} represents inverse cumulative probability function of standard normal distribution.

$$BNS = |F^{-1}(tpr) - F^{-1}(fpr)| \quad (1)$$

$$True\ \ positive\ rate\ (TPR) = \frac{TP}{TP + FN} \quad (2)$$

$$False\ Positive\ rate(FPR) = \frac{FP}{TN + FP} \quad (3)$$

Value of F^{-1} is determined from the Z-Score statistical table. F^{-1} values of both *true positive rate (tpr)* as well as *false positive rate (fpr)* are computed. Then the absolute differences of these values are considered as the BNS score. Here, *FP* denotes false positives i.e. benign files misclassified as malware and *TP* (true positives) denotes the correctly classified malware instances. *FN* (false negatives) gives malware misclassified as benign. *TN* (true negatives) is the correctly identified benign samples.

Mutual Information (MI) measures mutual dependence of two random variables. If we consider *f* as the features and *c* as the target class, *MI* furnishes the knowledge about the extent to which a feature *f* decreases the uncertainty in identifying the class *c*.

$$MI(f,c) = \sum_{c\in\{M,B\}} \sum_{f} \mathrm{P}(f,c) \log\left(\frac{\mathrm{P}(f,c)}{\mathrm{P}(f)\,\mathrm{P}(\mathrm{c})}\right) \qquad (4)$$

where, *P(f,c)* is the joint probability distribution function, *P(f)* and *P(c)* are the marginal probability distributions of variables *f* and *c*.

Bi-Normal Separation (BNS) and *Mutual Information (MI)* feature selection techniques are applied to the top 44 common *(M∩B)* features in the training samples. Experiments are performed on ranked features obtained with feature selection method with variable feature length. Later, we arrange the common feature list (malware and benign) in descending order of their BNS scores. From this list the top 30, 25, 20, 18 and 15 features are selected respectively to generate classification model which is used to predict future instances. Similar experiments are performed by selecting bottom 15, 18, 20, 25 and 30 BNS attributes. Classification models are constructed using classifier algorithms (Naïve Bayes [11], AdaBoost M1 with J48 [6] as base classifier, J48, Random forest [7] and IBK-5) implemented in WEKA [12].

F. Evaluation Metrics

Various evaluation metrics [10] like accuracy, precision, recall, F-measure etc. are calculated in order to predict unknown samples**.**

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \qquad (5)$$

$$Recall = \frac{TP}{TP+FN} \qquad (6)$$

$$False\ Negative\ Rate(FNR) = \frac{FN}{TP+FN} \qquad (7)$$

$$Precision = \frac{TP}{TP+FP} \qquad (8)$$

$$F-measure = \frac{2\times Recall \times Precision}{Recall + Precision} \qquad (9)$$

Our study gives answers to the following investigations:

1. Determination of *suitable feature category* (union, intersection, discriminant-malware and discriminant-benign feature) which results in higher accuracy.
2. Identification of the *best classifier* that provided higher accuracy with minimum false alarms.
3. Whether top/bottom *Bi-Normal Separation* features resulted in higher classification accuracy.
4. Determination of *optimal feature vector length* (30, 25, 20, 18 or 15) and robust feature selection method which results in higher accuracy.
5. Identification of *permissions* that are predominantly used by malware and benign specimens.

V. EXPERIMENT AND RESULTS

Experiments are conducted on a computer system with Ubuntu 12.04 operating system, Intel Core i3 CPU and 4G RAM. A total of 158 and 141 unique permissions are obtained from benign and malware training samples. Initially, we filtered top 40% of the permissions (indicate permissions that are required by at least 40% of the entire samples). This was performed as a part of initial preprocessing so as to eliminate sparse attributes (known as rare attributes).

From the reduced feature set, 63 benign and 56 malware features which constituted to 40% of the total permissions are extracted. Likewise, 44 permissions common to malware and benign apps (intersection of features), union of permissions in both benign and malware apps (75 permissions), 12 discriminant malware attributes and 19 discriminant benign permissions are respectively used to develop malware and benign models. To further prune the feature set, 18 features which have high *inter-class variance* are selected from the 44 common features (by eliminating non-prominent features; a few shown in Figure 3). The difference between the percentage of occurrence of each feature in the benign and malware training set are determined, and selected the features with greater difference. A feature vector table (FVT) is generated using these 18 features for developing training model which is further used for predicting unseen samples.

Figure 2 depicts differences in the permission frequencies (18 selected features amongst top 44 common features) and Figure 3 shows 20 features out of the eliminated features. Since there are variations in the permission frequencies in both malware and benign set, hence, we infer that permission can be used as the attributes for initial classification of samples.

From the 75 union features, the prominence of each feature in both benign and malware files are ascertained. Permissions found to be prominent in a particular class (benign or malware class) are finally used for identifying a target class. Thus, two independent feature lists are obtained one for malware and another for benign class. Subsequently, the feature space was reduced for both classes to contain 26 prominent malware and 49 benign attributes respectively. Similarly, FVT's were also generated for discriminant malware and discriminant benign features. Finally, a classification model is generated with the algorithm implemented in WEKA.

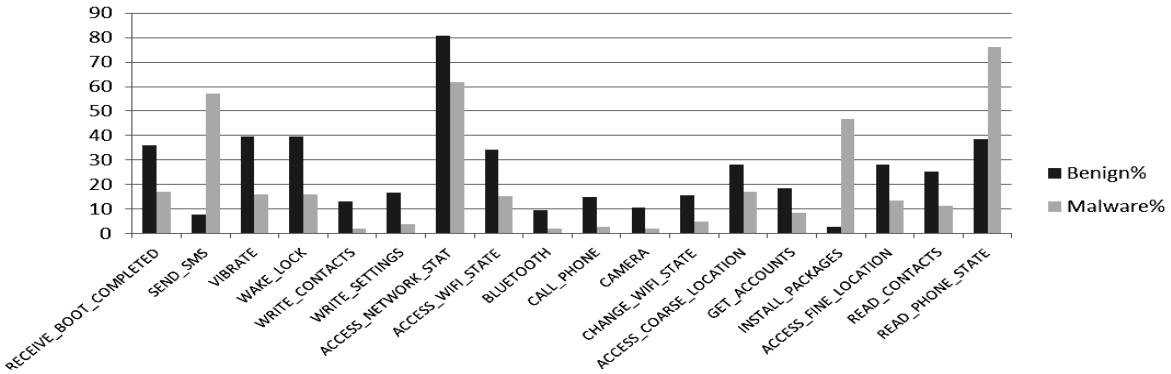

Fig. 2. Difference in the frequencies of 18 selected permission in malware and benign .apk files.

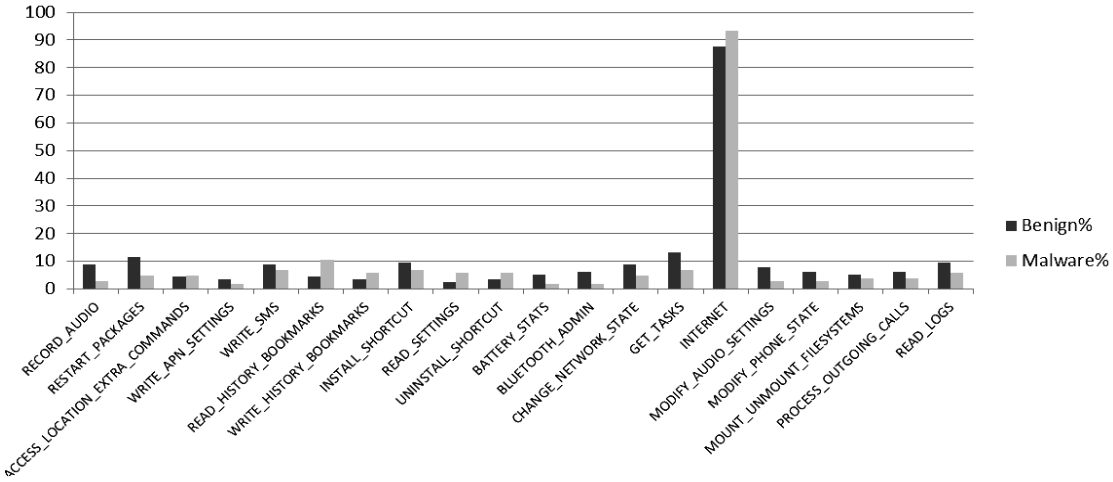

Fig. 3. Difference in the frequencies of 20 eliminated permissions in the malware and benign .apk files.

Finally, we aim at answering the questions presented in Section IV. The first two issues are to find out the robust features and classifier for classification. Common features are better compared to all other features. The best classifier is observed to be Random forest resulting in an accuracy of 82.48 % followed by the *k*-neighborhood (IBK-5) classifier with 82.02% accuracy. Table I shows the classification results based on the models built for the four above mentioned categories of feature lists.

In most cases, the bottom 20 features gave better accuracy compared to top BNS features (refer Table V). We observed that the top BNS (refer Table IV) features depicted less variation between the classes. Optimal feature vector length is found to be 15 (refer Table VII) when Mutual Information is used to synthesize original feature space.

TABLE I
PERFORMANCE MEASURES OF THE FOUR CATEGORIES OF FEATURE LISTS

Feature Categories	**M ∩ B**		**M ∪ B**				**B \ M**		**M \ B**	
Original Feature Length	**44**		**75**				**19**		**12**	
Reduced feature length	**18**		**49**		**26**		**19**		**12**	
Evaluation Measure	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
NB	**75.11**	0.747	66.35	0.702	69.12	0.645	64.05	0.723	71.88	0.634
IBK-5	**82.02**	0.807	70.50	0.750	70.96	0.651	60.36	0.720	71.42	0.626
Adaboost M1(J48)	**80.64**	0.807	69.58	0.722	71.42	0.655	47.93	nd	71.42	0.626
J48	**80.18**	0.799	70.50	0.741	73.27	0.666	58.98	0.712	69.12	0.582
RF	**82.48**	0.806	76.03	0.763	72.35	0.659	63.59	0.718	71.42	0.634

TABLE II
PROMINENT FEATURES AND THEIR MEANING

Permissions	Meaning
WRITE_EXTERNAL_STORAGE	Permission for an application to write to the external storage
READ_PHONE_STATE	Permission read only access to phone state
VIBRATE	Allows access to the vibrator.
WAKE_LOCK	Allows using PowerManager WakeLocks to keep processor from sleeping or screen from dimming
ACCESS_NETWORK_STATE	Permission to access network information
RECEIVE_BOOT_COMPLETED	Allows an application to receive ACTION_BOOT_COMPLETED that is broadcasted after booting.
SEND_SMS	Allows an app to send SMS
ACCESS_WIFI_STATE	Permission for the app to access network information
ACCESS_COARSE_LOCATION	Permission for the app to access approximate location by means of towers and wi-fi
ACCESS_FINE_LOCATION	Permission for the app to access precise location by means of towers and wi-fi
READ_CONTACTS	To read contact list of the device's user
INSTALL_PACKAGES	Permission for package installation
RECEIVE_SMS	Permission for the app to check incoming SMS to save and process them.
GET_ACCOUNTS	Permission to access Account service list
READ_SMS	Permission of an app to read SMS

Table II lists the prominent features used by the malware dataset. From the above tabulated results (Table I to VII), we infer the following:

- As we increase the feature length redundant features begin to appear which leads to increased misclassification.

TABLE III
TRIVIAL FEATURES AND THEIR MEANING

Permissions	Meaning
ACCESS_LOCATION_EXTRA_COMMANDS	Permission for an application to access extra location provider commands.
MODIFY_AUDIO_SETTINGS	Permission for an application to modify audio settings.
MODIFY_PHONE_STATE	Permission for an application to modify mmi, phone state - power on etc.
MOUNT_UNMOUNT_FILESYSTEMS	Permission for an application to mount and unmount filesystem for removable storage
PROCESS_OUTGOING_CALLS	Allows an application to process outgoing calls .
READ_LOGS	Permission for an application to read system log files.
WRITE_APN_SETTINGS	Permission for an application to write apn settings.
RECORD_AUDIO	Permission for an application to record audio.
WRITE_HISTORY_BOOKMARKS	Permission for an app to write user's bookmarks .
INSTALL_SHORTCUT	Permission for an app to install a shortcut in launcher.

- Bottom BNS features resulted in better accuracy in comparison to top BNS features. This is due to the fact that on observing the feature vectors the frequency of top BNS features were nearly same in both the classes. Also, many features were almost rare in both data sets.
- MI feature selection method resulted in better identification of unknown samples in comparison to BNS feature selection technique (refer Table VI and VII). This is due to the fact that prominent attributes having high correlation with target class is obtained using MI compared to BNS, and minimum correlating features are eliminated.

TABLE IV
EVALUATION PARAMETERS FOR TOP BNS FEATURES

Feature Length	Top BNS features									
	30		25		20		18		15	
Evaluation Measure	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
NB	67.71	0.726	63.13	0.701	63.59	0.708	64.51	0.720	57.60	0.693
IBK-5	61.29	0.714	86.63	0.879	58.52	0.701	62.21	0.720	54.37	0.689
Adaboost M1 (J48)	68.20	0.718	61.29	0.695	60.82	0.701	47.90	n.d	47.92	n.d
J48	70.04	0.747	62.21	0.715	61.29	0.714	61.29	0.714	47.92	n.d
RF	69.12	0.735	61.75	0.684	63.59	0.708	66.20	0.729	58.52	0.7

TABLE V
EVALUATION PARAMETERS FOR BOTTOM BNS FEATURES

Feature Length	Bottom BNS features									
	30		25		20		18		15	
Evaluation Measure	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
NB	72.35	0.734	72.81	0.735	**73.27**	0.743	70.50	0.714	73.27	0.715
IBK-5	80.64	0.803	82.02	0.811	**78.80**	0.778	77.88	0.766	79.72	0.784
Adaboost M1(J48)	81.56	0.811	80.18	0.798	**80.18**	0.794	80.18	0.796	79.70	0.794
J48	80.18	0.794	80.18	0.794	**82.48**	0.820	79.70	0.805	76.95	0.766
RF	**80.18**	**0.796**	**80.18**	**0.798**	**81.10**	**0.814**	**80.18**	**0.798**	**80.64**	**0.809**

TABLE VI
EVALUATION PARAMETERS FOR MUTUAL INFORMATION FEATURES

Feature Length	Mutual Information features									
	15		18		20		25		30	
Evaluation Measure	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
NB	71.42	0.720	73.73	0.744	73.27	0.741	73.27	0.745	70.96	0.727
IBK-5	77.88	0.785	79.35	0.792	79.72	0.796	79.72	0.796	79.76	0.798
Adaboost M1 (J48)	80.18	0.808	78.34	0.792	77.88	0.783	79.72	0.801	81.10	0.809
J48	74.56	0.763	76.95	0.776	76.03	0.779	78.80	0.788	80.64	0.798
RF(40) Seed 4	**81.56**	**0.814**	79.72	0.8	76.95	0.782	79.26	0.796	80.64	0.803

- Better results are obtained using Random Forest as it is an ensemble based classifier. RF assigns a class to the samples by aggregating the results from multiple classifiers.
- Permissions with similar frequency distribution in benign and malware are considered as trivial features for classification (refer Figure 3 and Table III) and are eliminated by the feature selection methods implemented in this work thereby resulting in higher accuracy.

TABLE VII
COMPARATIVE ANALYSIS OF BI-NORMAL SEPARATION AND MUTUAL INFORMATION FEATURE SELECTION METHOD

Feature Length	BNS(15)		MI(15)	
Evaluation Measure	**Acc.**	**F1**	**Acc.**	**F1**
NB	72.35	0.732	71.42	0.720
IBK-5	78.80	0.783	77.88	0.785
Adaboost M1(J48)	78.34	0.791	80.18	0.808
J48	74.65	0.763	74.65	0.763
RF(40) Seed 4	80.64	0.809	**81.56**	**0.814**

VI. CONCLUSION

The proposed malware analyzer Droid Permission Miner, uses static feature extraction methodology. The features extracted are permissions from manifest file. We observed that better classification accuracy is obtained with *Mutual Information* feature selection methods with a small feature length of 15 features. Our proposed method can be used for initial classification of malware or benign specimens. To obtain higher classification accuracy in future, we would like to create an optimal feature vector by combining features such as Dalvik opcode, permissions, critical system calls and Java reflection**.**

REFERENCES

[1]. Androguard. http://code.google.com/p/androguard/.

[2]. Malware apk : http://contagiominidump.blogspot.in/2011/07/take-sample-leave-sample-mobile-malware.html

[3]. Apk file format. http://www.file-extensions.org/article/android-apk-file-format-description

[4]. Z-Score Table : http://www.stat.tamu.edu/~lzhou/stat302 /standardnormaltable.pdf

[5]. R.Battiti, "Using Mutual Information for Selecting Features in Supervised Neural Net Learning" IEEE Transactions On Neural Networks, Vol. 5, No. 4, July 1994

[6]. Y. Freund and R.E Schapire, "Experiments with a new Boosting Algorithm, Machine Learning" *in Proceedings of the Thirteenth International Conference*, 148–156, 1996

[7]. A.Liaw and M. Wiener, "Classification and Regression by Random Forest", R News, 2/3:18-22, December, 2002.

[8]. G. Forman. "An Extensive Empirical Study of Feature Selection Metrics for Text Classification". *Special Issue on Variable and Feature Selection, Journal of Machine Learning Research*, 3(Mar):1289-1305, 2003.

[9]. G. Forman, "BNS Scaling: A Complement to Feature Selection for SVM Text Classification". Hewlett-Packard Labs Tech Report HPL-2006-19, 2006.

[10]. Tan, Pang-Ning and Steinbach, Michael and Kumar, Vipin, Introduction to Data Mining, First Edition, 2005, Addison-Wesley Longman Publishing Co., Inc., Boston, MA USA.

[11]. J. Chen, H. Huang, S. Tian, Y. Qu, "Feature Selection for Text Classification with Naive Bayes", Expert Systems with Applications 36 (2009) 5432–5435.

[12]. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, Ian H. Witten, "The WEKA Data Mining Software: An Update", SIGKDD Explorations, Volume 11, Issue 1,2009

[13]. T. Blasing, L. Batyuk , A. D. Schmidt , S. A. Camtepe, and S. Albayrak, "An Android Application Sandbox System for Suspicious Software Detection"2010.

[14]. A. Shabtai and Y. Elovici, "Applying Behavioral Detection on Android-Based Devices". In the proceedings of Third International Conference, Mobilware 2010, Chicago, IL, USA, June 30 - July 2, 2010, pp.235-249

[15]. Shabtai A., "Malware Detection on Mobile Devices", *In the Proceedings of 11th International Conference on Mobile Data Management, 2010.*

[16]. T. Isohara, K. Takemori and A. Kubota,"Kernel-based Behavior Analysis for Android Malware Detection",2011.

[17]. A.P. Felt, M. Finifter, E. Chin, S. Hanna and D. Wagner, "A Survey of Mobile Malware in the Wild", *In the proceedings of SPSM11* , October 17,2011.

[18]. V. Manjunath, "Reverse Engineering of Malware on Android", 2011,www.sans.org/reading-room/whitepapers/

[19]. D. A. Heger, "Mobile Devices - An Introduction to the Android Operating Environment Design, Architecture, and Performance Implications",2011.

[20]. D. J. Wu, C. H. Mao, T. E. Wei, H. M. Lee1, K. P. Wu. "DroidMat: Android Malware Detection through Manifest and API Calls Tracing", *In Proceedings of Seventh Asia Joint Conference on Information Security*, 2012, pp.62-70

[21]. A.Shabtai, U. Kanonov, Y. Elovici, C. Glezer and Y. Weiss, "Andromaly: A Behavioral Malware Detection Framework for Android Devices", J. Intell. Inf. Syst. 38, 1 February, 2012.

[22]. G. Dini1, F. Martinelli, A. Saracino and D. Sgandurra, "MADAM: a Multi-Level Anomaly Detector for Android Malware". *In the proceedings of 6th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security, MMM-ACNS 2012, St. Petersburg, Russia,* October 17-19, 2012, pp.240-253

[23]. Y. Aafer, W. Du and H. Yin, "DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android", 2013.