Mining and Detection of Android Malware Based on Permissions

Abdirashid Ahmed Sahal
Dept. of Computer Engineering
Gebze Technical University
Kocaeli, Turkey
asahal@gtu.edu.tr

Shahid Alam
Dept. of Computer Engineering
Adana Science and Technology University
Adana, Turkey
salam@adanabtu.edu.tr

Ibrahim Soğukpınar
Dept. of Computer Engineering
Gebze Technical University
Kocaeli, Turkey
ispinar@gtu.edu.tr

***Abstract*—Due to the open app distribution and more than two billion active users, Android platform continues to serve as low-hanging fruit for malware developers. According to the McAfee threat report, the number of malware families found in the Google Play increased by 30% in 2017. Permission-based access control model is one of the most important mechanisms to protect Android apps against malware. In this paper, we propose a new permission-based model that enhances the efficiency and accuracy of Android malware analysis and detection, and has the capability of potentially detecting previously unknown malware. In this new model, we improve the feature selection by introducing a new weighting method, named TF-IDFCF, based on the class frequency (CF) of the feature. The results of our experiments show that our proposed method has a detection rate of greater than 95.3% with a low false positive rate, when tested with different classifiers.**

***Index Terms*—Android, Permissions, Malware Analysis and Detection, TF-IDF, Machine Learning.**

I. Introduction

Recently, Android has become the most selling operating system on mobile devices [1]. Android OS monthly has over two billion active users. Malware writers are actively and continuously developing malware programs to target Android platform. This continuous evolution and the diversity of malware pose a major threat to Android applications. According to the McAfee threat report, number of malware families found in the Google play increased by 30% in 2017 [2]. Different solutions have been proposed to protect mobile users from the increasing threats of Android malware. Permission-based access control model is the most important mechanism for Android protection against malware apps. In this paper, we use multiple machine learning algorithms with permission datasets to build and train models to classify Android malicious apps. We improved the Term Frequency-Inverse Document Frequency (TF-IDF) method, introduced new feature selection method that increases the detection accuracy. In this paper, we propose a new permission-based static analysis framework for the classification of Android applications into benign and malware. We improve upon other permission-based approaches by introducing a feature selection method based on TF-IDF. This method improves the efficiency of malware analysis and detection, and obtains a high accuracy. The results of our experiments show that our proposed framework has a detection rate of more than 95.3% using most of the basic classifiers, such as SVM, J48, Naive Bayes and KNN. Our contribution of this paper is an improvement of TF-IDF weighting on vector space model. The TF-IDF method considers both TF and IDF [3]. If the TF is high and the term only appears in some part of the applications, then this term has a very good ability to differentiate the applications. A feature occurring frequently in the applications within same class represents more characteristics of the class. Therefore, we use total feature occurrence as a new parameter and enhance the TF-IDF to improve the efficiency of our classifier.

The remaining parts of the paper are organized as follows. In section 2 related works are presented. The proposed model is introduced in section 3. In section 4 experimental results are given. The last section is conclusions.

II. Related work

Many research has been performed on Android malware classification, using permission related features. We highlight some of related works of permission-based malware detection.

X. Liu and J. Liu proposed a framework that considers both requested and used permissions in the Android applications [4]. This framework is two layered malware detections and uses machine learning techniques to get high detection accuracy with the potential of detecting Android malware applications based on permissions. P. Rovelli and Ý. Vigfússon proposed a simple, client-server architecture malware detection system based on permissions which views requested permissions as behavioral markers [5]. The system has the server-side and client-side permission checker parts. The client-side part extracts the permissions from the Android apps and forwards the extracted permissions to the server-side part. The server-side part classifies the application as benign or malware.

978-1-5386-7893-0/18/$31.00 ©2018 IEEE

Z. Aung and W. Zaw proposed machine learning based framework for malicious apps detection and security enhancement of Android based mobile device users [6]. This framework monitors different permission-based features extracted from Android applications and uses machine learning classifies to tag the application as benign or malicious. The system uses information gain as feature selection method. They extracted more permissions from different downloaded applications from Android markets to generate the model. Z. Xiaoyan, F. Juan proposed a malware detection system that analyzes the application activity of the mobile phone. [7]. Extracts features, uses principal component analysis method as feature selection and applies support vector machine method to train and build classifier for malware detection. With the contributions of the android based mobile device users, this framework has the ability to classify the android apps as benign and malicious. D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens proposed a lightweight Android malware detection approach that enables detecting malware applications in the smartphones directly [8]. The approach performs broad static analysis by collecting all possible features of Android APK. It embeds in a joint vector space the whole features and automatically detects the malicious apps. The approach gives the advantage of time efficiency in analyzing unknown apps. The approach can be used on the mobile phone directly, and enables protection of installation from untrusted sources. Y. Aafer, W. Du, and H. Yin, have performed a study thorough malware analysis for extracting malware relevant features and behavior captured at API level, they trained multiple robust and lightweight machine learning classifiers using the generated dataset from the extracted feature set [9]. Their experiment result using KNN classifier shows a TPR of 99% and 2.2%.

III. Overview of the system

In this section, we highlight the preprocessing model, feature selection and the learning methods we used in the proposed framework. Figure 1 shows the overview of the proposed framework.

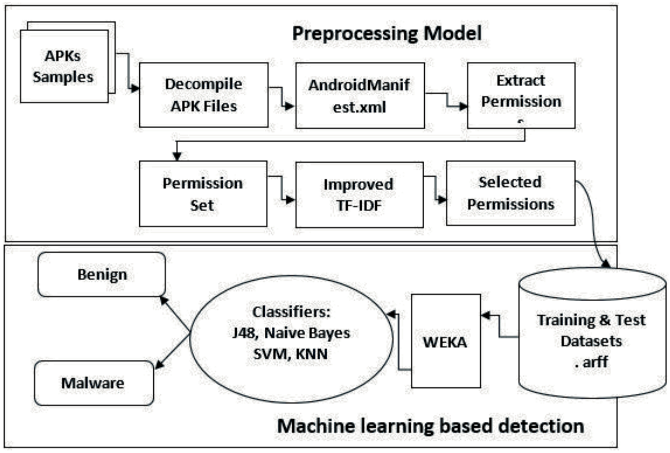

Fig. 1. Overview of the system

A. Preprocessing Model

In this stage, to get the resources we decompile the Android APK sample files using the APKTool. We extract the permissions that the Android applications requested from their AndroidManifest.xml file. After extracting the permissions, we identify the unique permissions in malware and benign then we build a binary matrix storing (0,1) binary values for the features, and then we applied improved TF-IDF weighting method for feature selection to build our datasets.

B. Feature selection

The TF-IDF weighting method in vector space model is often used in information retrieval and categorization, gives more weight only to the term that appears rarely in an application.

$$w_i = tf_i \times log(\frac{N}{n_i}) \quad (1)$$

In equation (1), tf_i represents the raw frequency of the term i, N is the total number of applications in the corpus (corpus means the complete dataset), and n_i is the number of applications that feature i appears in. We know some features can have different weights in different classes, so we improved the TF-IDF method to represent more class's characteristics, by considering the feature frequently appearing in the applications within the same class.

$$w_i = tf_i \times log(\frac{n}{n_i}) \times c_i \quad (2)$$

In equation (2), we use a class-based TF-IDF, instead of corpus-based method, tf_i is the raw frequency of the feature i, n is the total number of applications in the class, and n_i is the number of applications in the class that feature i appears in. To give priority to the feature that appears more frequently in the same class, we introduced another term c_i which is the class frequency of feature i, and hence this new equation/model also includes class's characteristics.

In this stage, we collect all possible features extracted from the Android APK files. We use the new TF-IDF based method, named TF-IDF-CF shown in equation (2), to select best features and give them weights in order to enhance the detection accuracy of our proposed framework, and to make the learning easy. After that the selected features as binary values are stored as a feature vector. Following is an example of the benign and malware feature vectors extracted from an Android APK file:

1, 1, 1, 1, 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,

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, 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

(a) Benign

0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 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, 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, 0, 0, 0, 0, 0, 0

(b) Malware

C. Model Learning and Generation

In this stage, in order to detect automatically potential malicious applications, we trained multiple classifiers using the following machine learning algorithms: J48 decision tree, Naive Bayes, SVM and KNN.

We used the J48 Ross Quinlan's implementation of C4.5 Decision Tree-based learning algorithm.J48 [10] decision tree is an important type of algorithm that builds classification model in the form of a tree. J48 is a machine-learning model based on the prediction that makes a decision tree which is based on the attribute values of the training data for classifying a new item. Naive Bayes [11] is one of the supervised classification methods and a powerful algorithm for predictive modeling. The classification techniques of Naive Bayes are probabilistic algorithm based on Bayes' Theorem with an assumption of independence among predictors. Naive Bayes is an easy model to build and works well in large datasets. It observes the whole attributes separately and makes their usage in the data simpler. SVM [12] primarily a classier method based on decision planes concept. In machine learning Support Vector Machines are one of the most popular algorithms. SVMs can efficiently perform linear and non-linear classifications. The KNN [13] is a non-parametric classification method. It is very effective and simple classification algorithm which is fundamentally different from the other learning algorithms. KNN mostly referred lazy learner because of memorizing from the training dataset.

IV. Classification and Evaluation

A. Dataset

We have 1000 APK samples (500 malware and 500 benign) collected from different sources. The benign samples are collected from the Google Play Store [14]. The sample of malicious apps we took from Contagio Mobile [15] and Android Malware Genome Project [16]. We decompiled, extracted the features and, created training and testing datasets of 700 and 300 samples respectively. Total 385 features were selected to enhance the accuracy.

TABLE I
Datasets

	Samples	Malware	Benign	Attributes
Training	700	350	350	385
Testing	300	150	150	385
Total	1000	500	500	385

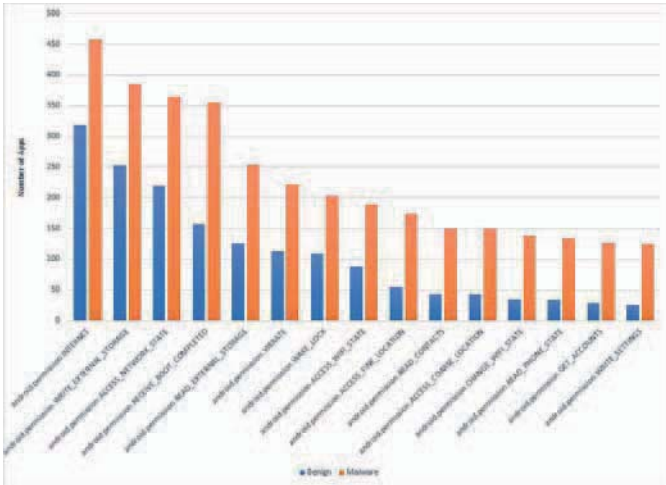

Fig. 2. Frequency of specific permissions (15 shown here) in 500 malwares and 500 benign APKs

B. Permissions and Security

Permission is a special privilege that allows applications to access sensitive media or hardware feature in a smartphone. Certain permissions (such as, Camera and Location, etc.) should be given with greater care. The permissions required by an Android application is declared in its Android manifest file, and it can also define some additional permissions to use for restricting some components. During the runtime an Android application asked from the user to allow (permit) certain actions.

For security and privacy purposes, during installation Android System gives each application a unique user and group ID. In this way the data for each application remains private, and the other applications have no access to this data. The only exception is where a user gives permissions to other applications to access this data, such as contacts, media files etc. Number of permissions asked in the 500 malwares and 500 benign samples are shown in Figure 2. Here we discuss some important permissions that can be used to perform malicious actions:

WRITE_SMS, SEND_SMS and READ_SMS: These permissions allow an application to gain access to harmful API calls, and hence allowing the application to read, write, and send SMS (such as contacts, banking information etc.) without notifying the user. CALL_PHONE: This permission allows an application to make calls on behalf of a user without confirming from the user, and hence is dangerous to the privacy of a user. READ_HISTORY_BOOKMARKS: This permission allows an application to read a user's browsing history and bookmarks, and hence is a risk to the privacy of a user. ACCESS_COARSE_LOCATION and ACCESS_FINE_LOCATION: These permissions allow an application to access the coarse (e.g., Cell-ID, Wi-Fi) and fine (e.g., GPS) locations citesarma2012android of a smartphone, and hence pose a risk to the privacy of a user.

C. Classification Models

In this section, we evaluated using Weka tool the classification capability of our framework for Android APK files as benign or malware on two datasets, a training set of 700 and a testing set of 300 samples. This framework gets the original feature set by extracting permissions from Android APK files using reverse engineering tool and forms the feature set by using the newly introduced method named TF-IDF-CF. We used the techniques of machine learning to classify the Android applications. We used multiple learning algorithms such as J48, Naive Bayes, SVM and KNN to test our model. The metrics we used are as follows:

TP (True Positive): Correctly detected number of malware apps. **FP** (False Positive): Number of benign applications wrongly identified as malware. **TN** (True Negative): Correctly detected number of benign apps. **FN** (False Negative): Number of malware applications wrongly identified as benign.

We used the below metrices to evaluate the performance of our proposed permission-based detection framework.

True Positive Rate (TPR): Number of samples correctly classified as malware out of the total malware dataset (TP / TP+FN). **False Positive Rate (FPR)**: Number of samples wrongly classified as malware out of the total benign dataset (FP / TN+FP). **Overall Accuracy (ACC)**: Proportion of Android apps, that are correctly identified as either malicious or benign app. (TP+TN / TP+TN+FP+FN).

D. Evaluation

To check the accuracy of our permission-based detection method we also used dataset containing 1000 APK samples, including 500 malwares and 500 benign samples, with 10-fold cross-validation method. This method divides the dataset into ten parts and, takes one part the 10% as testing set and the rest 90% as training test.

To further evaluate and make a comparison with our proposed technique, we selected two other well-known information mining techniques, Principal Component Analysis (PCA) [17] and Information Gain [3]. Both these techniques reduce the number of attributes to achieve a high detection rate. PCA is a statistical method used to emphasize variation and bring out strong patterns in a dataset. Information Gain depends on the entropy of an attribute and selects a feature that provides the foremost information gain. Using the same dataset, we compared our method to Information Gain and PCA. Results are shown in table III. As seen in table III the use of our proposed method for assigning weights to each feature extracted from the APK has higher accuracy rate than the other two techniques.

TABLE II
EXPERIMENTAL RESULTS

Dataset	Method	TPR	FPR	ACC
Training	J48	0.991 %	0.009 %	99.1429 %
Training	Naive Bayes	0.98 %	0.02 %	98 %
Training	SVM	1 %	0 %	100 %
Training	KNN	1 %	0 %	100 %
Test	J48	0.97 %	0.03 %	97 %
Test	Naive Bayes	0.97 %	0.03 %	97 %
Test	SVM	0.983 %	0.017 %	98.3333 %
Test	KNN	0.953 %	0.047 %	95.3333 %

TABLE III
COMPARED RESULTS

Information Gain Method				
Cross Validation	Classifier	TPR	FPR	ACC
10-fold	J48	0.86 %	0.14 %	86 %
10-fold	Naive Bayes	0.819 %	0.181 %	81.9 %
10-fold	SVM	0.864 %	0.136 %	86.4 %
10-fold	KNN	0.856 %	0.144 %	85.6 %
Principal Component Analysis Method				
Cross Validation	Classifier	TPR	FPR	ACC
10-fold	J48	0.864 %	0.136 %	86.4 %
10-fold	Naive Bayes	0.578 %	0.422 %	57.8 %
10-fold	SVM	0.815 %	0.185 %	81.5 %
10-fold	KNN	0.843 %	0.157 %	84.3 %
Our Proposed (TF-IDF-CF) Method				
Cross Validation	Classifier	TPR	FPR	ACC
10-fold	J48	0.975 %	0.025 %	97.5 %
10-fold	Naive Bayes	0.973 %	0.027 %	97.3 %
10-fold	SVM	0.986 %	0.014 %	98.6 %
10-fold	KNN	0.968 %	0.032 %	96.8 %

E. Limitations

In this work, our weight assigning method relies on the labeled (into classes) data. Before training, the data needs to be divided into classes, which enables us to calculate the class frequency (c_i) and allows us to use class-based instead of corpus based TF-IDF. Currently the only classes data is divided into, before training, are malware and benign.

V. Conclusions

Permission is one of the most important features in Android security, and meaningful in malware detection. Our proposed permission-based framework uses machine learning algorithms to detect potentially malware apps. Also, to improve the efficiency of permission-based Android malware analysis and detection we introduced a new method based on TF-IDF to assign weight to each feature extracted from an Android APK file. We evaluated the new technique using different metrics and achieved a detection rate higher than 95.3% using different classifier algorithms. To further improve malware detection, in future we will add more features, such as, API calls etc., to our proposed framework. To improve the weight assigning method, in future, we are going to divide the data (especially malware samples) into more (than two) classes based on similarity measures.

Acknowledgements

This work was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK), Grant No: ARDEB-116E624.

References

[1] "The Statistics Portal: Statistics and studies from more than 22,500 sources," https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/, accessed: 2018-05-01.

[2] "McAfee mobile threat report Q1," https://www.mcafee.com/enterprise/en-us/assets/reports/rp-mobile-threat-report-2018.pdf, accessed: 2018-05-29.

[3] D. M. Christopher, R. Prabhakar, and S. Hinrich, *Introduction to information retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[4] X. Liu and J. Liu, "A two-layered permission-based android malware detection scheme," in *Mobile cloud computing, services, and engineering (mobilecloud), 2014 2nd ieee international conference on*. IEEE, 2014, pp. 142–148.

[5] P. Rovelli and Ý. Vigfússon, "Pmds: Permission-based malware detection system," in *International Conference on Information Systems Security*. Springer, 2014, pp. 338–357.

[6] Z. Aung and W. Zaw, "Permission-based android malware detection," *International Journal of Scientific & Technology Research*, vol. 2, no. 3, pp. 228–234, 2013.

[7] Z. Xiaoyan, F. Juan, and W. Xiujuan, "Android malware detection based on permissions," *The Institute of Engineering and Technology*, 2014.

[8] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket." in *Ndss*, vol. 14, 2014, pp. 23–26.

[9] Y. Aafer, W. Du, and H. Yin, "Droidapiminer: Mining api-level features for robust malware detection in android," in *International conference on security and privacy in communication systems*. Springer, 2013, pp. 86–103.

[10] N. Bhargava, G. Sharma, R. Bhargava, and M. Mathuria, "Decision tree analysis on j48 algorithm for data mining," *Proceedings of International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 6, 2013.

[11] T. R. Patil and S. Sherekar, "Performance analysis of naive bayes and j48 classification algorithm for data classification," *International Journal of Computer Science and Applications*, vol. 6, no. 2, pp. 256–261, 2013.

[12] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[13] M.-L. Zhang and Z.-H. Zhou, "Ml-knn: A lazy learning approach to multi-label learning," *Pattern recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.

[14] "Google Play Store google play store," https://play.google.com/store/apps, accessed: 2018-05-01.

[15] "Contagio Mobile genome project," http://contagiominidump.blogspot.com.tr/, accessed: 2018-05-01.

[16] Y. Zhou and X. Jiang, "Android malware genome project," *Disponibile a http://www. malgenomeproject. org*, 2012.

[17] I. T. Jolliffe, *Principal Component Analysis*. New York, NY, USA: Springer-Verlag, 2002.