International Journal of Engineering & Technology, 7 (4.31) (2018) 311-315

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Evaluation of Feature Selection Algorithm for Android Malware Detection

Nurul Hidayah Mazlan[1], Isredza Rahmi A Hamid[2]*

Information Security Interest Group (ISIG)
Faculty of Computer Science & Information Technology,
Universiti Tun Hussein Onn Malaysia, Johor Malaysia
**Corresponding author E-mail: rahmi@uthm.edu.my*

Abstract

This paper synthesizes an evaluation of feature selection algorithm by utilizing Term Frequency Inverse Document Frequency (TF-IDF) as the main algorithm in Android malware detection. The Android features were filtered before detection process using TF-IDF algorithm. However, IDF is unaware to the training class labels and give incorrect weight value to some features. Therefore, the proposed approach modified the TF-IDF algorithm, where the algorithm focused on both sample and feature. Proposed algorithm applied considers the feature based on its level of importance. The related best features in the sample are selected using weight and priority ranking process. This increases the effect of important malware features selected in the Android application sample. These experiments are conducted on a sample collected from DREBIN dataset. The comparison between existing TF-IDF algorithm and modified TF-IDF (MTF-IDF) algorithm have been tested in various conditions such as different number of sample, different number of feature and combination of different types of feature. The analysis results show feature selection using MTF-IDF can improve malware detection analysis. MTF-IDF proved either using various kinds of feature or various kinds of dataset size, algorithm still effective for Android malware detection. MTF-IDF algorithm also proved that it could give appropriate scaling for all features in analyzing Android malware detection.

Keywords: *Android malware; Detection; Term Frequency-Inverse Document Frequency (TF-IDF).*

1. Introduction

Android malware detection has drawn a lot concern for many researchers. Up until now, Android malware has become more complex where the attacker can divert the Android malware detection techniques. Existing Android malware detection is based on static and dynamic analysis [1]. Static analysis is a fast and efficient method. However, static analysis approach prone to high false negative rates due to evolution in code basis and code repacking. Dynamic analysis aims to provide effective and efficient methods to extract unique patterns of each malware family based on its behavior. Although to enhance malware detection, dynamic analysis required more features, such as the system call and network activities. Therefore, it is crucial to identify Android malware based on feature selection approach. In this paper, we want to emphasize two research issues that are:

1. What is the best approach to detect android malware: Android malware detection is an essential aspect to combat Android malware. Even though, numbers of Android malware approach have been proposed in order to solve the Android malware detection problem. The number of Android malware keep increasing which shows that the attacker has become more advanced and can easily bypass the Android malware detection techniques. Thus, a new solution for Android malware detection is required where new feature selection approach by modifying the Term Frequency Inverse Document Frequency (TF-IDF) algorithm is proposed.
2. How to measure the feature selection approach: We consider to evaluate the effectiveness of the feature selection approach using accuracy, True Positive and False Positive Rate [2].

Section 2 describes Android malware detection related research regarding feature selection algorithm. Section 3 observes the data and feature set used in the experiment and discussed about TF-IDF algorithm and modified TF-IDF (MTF-IDF) as well. Section 4 discusses the performance of the feature selection algorithms and the effectiveness of the proposed feature selection approach. Finally, Section 5 concludes our work and projected direction for future work.

2. Related Work

This section will examine the TF-IDF algorithm and related work that use TF-IDF for data classification.

2.1. Feature Selection Algorithm

In analyzing Android malware features, huge number of sample involved and it can affect the algorithm effectiveness when noisy and irrelevant feature include in the dataset. Thus, feature selection has the ability in reducing noisy and irrelevant feature to a useful data could lead to more accurate results using machine learning algorithms [3]. Yerima et. al [4] proposed an approach that utilizes ensemble learning for Android malware detection. By combining advantages of static analysis with the efficiency and performance of ensemble machine learning, Android malware detection accuracy has been improved. The machine learning

models are built using a large repository of malware samples and benign applications from a leading antivirus vendor. This approach apply permission based feature in the experiment.
Latest work from Bhattacharya *et al* [5] proposed a framework that extracts the permission features of manifest files, and then generates feature vector. This research conducted using six different feature ranking tools. By comparing with previous work, this paper highlights new goal in Android malware detection. Previous paper focus on reduce classification time and improvement of the accuracy with different reduction tools. Note that this is an extension work of [6] where we use Application Program Interface (API) call based and permission based features to detect android malware.

2.2. Term Frequency-Inverse Document Frequency (TF-IDF) Algorithm in Detecting Malware for Android Platform

TF·IDF commonly used in text classification because it richer and more successful representation for retrieval and categorization purposes [7][8]. TF-IDF join term's frequency in the document (TF) and its frequency in the document's collection known as Document Frequency (DF). The term vector for each document is created where each cell in the vector represent TF as in Eq. 1. In term weighting, a vocabulary of the entire document collection is constructed. Each term in the document is characterized using its own frequency from the dataset called as DF shown in Eq.2. *N* is total number of documents in the dataset, while *DF* is number of files in which the terms show. Therefore, the term TF-IDF equation is defined in equation Eq. 3, where the weighted of specification behavior is calculated.

$$TF = \frac{term\_frequency}{\max(term\_frequency\_in\_document)} \quad (1)$$

$$IDF = \log\left(\frac{N}{DF}\right) \quad (2)$$

$$TF - IDF = TF \times IDF \quad (3)$$

TF-IDF formula has been changed by SVM-based approach [9] to compute the weight of android malware features. Eq. 4 is where, N_{ij} is the number of times that Android application, D_j calls specific android malware feature. M_j is total number of times, D_j calls all different android malware feature. The IDF_i in Eq. 5 is where the sum of malware in the training dataset, D and D_i is number of times the android malware feature is called.

$$TF_{ij} = \frac{N_{ij}}{M_j} \quad (4)$$

$$IDF_i = \log\left(\frac{D}{D_i + 1}\right) \quad (5)$$

3. Android Feature Selection Approach

The proposed feature selection approach for Android malware detection approach is discussed in this section.

3.1. Android Malware Detection Model

Android malware detection model consists of four types of process; preprocessing of raw Android data collected from DREBIN, feature extraction and selection, feature selection approach and classification using machine learning algorithm and finally the evaluation of the detection result as shown in Figure 1. The model is created based on data mining approach. We try to build a classifier for Android malware detection that can classify the android malware into malware or benign.

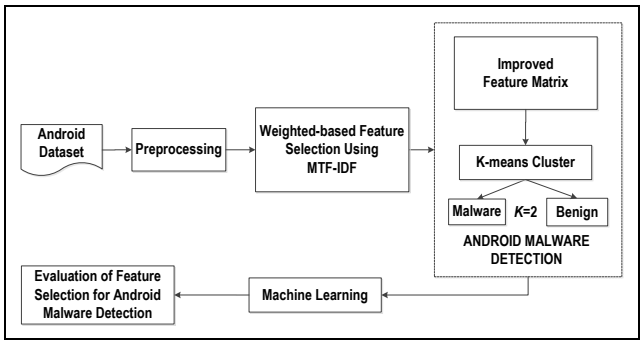

Fig. 1: Android Malware Detection Model

The Android malware detection model is assessed with DREBIN [10] dataset that consists of benign and malicious application. In this study, two types of Android features have been used, that are Application Program Interface (API) call (dynamic feature) and permission (static feature). The proposed feature selection algorithm aims to determine the feature matrix for detecting an Android malware feature. Android malware detection algorithm consists of weighted based feature selection and K-means clustering, which very important in this study. In the preprocessing process, all data are cleaned up and normalized. Then, all samples are extracted into human readable format in .xml form. Each incoming samples will perform the weighted based feature selection algorithm using TF-IDF and MTF-IDF. Then, K-means clustering algorithm is used to class the sample either into malware or benign. Lastly, the feature matrix is constructed and tested using machine learning algorithm in order to evaluate the performance of the proposed feature selection algorithm (MTF-IDF).
In evaluation phase, Random Forest [11], Bagging and Decision Table were used to test the accuracy of the feature selection method. These algorithms have the same capabilities on detecting missing class value, nominal class, numeric class and binary class. Random Forest [12] algorithms works derived from combination of tree predictors. By taking into account the voted classes of all individual trees, class with the highest vote is selected as the best output. Bagging [13] works by creates a number of base classifiers for its ensemble by training each base classifier using random redistribution of the training set. Decision table [12] classification algorithm builds a simple decision table majority classifier, and summarizes the dataset with a decision table which contains the same number of features as the original dataset.
We use Waikato Environment for Knowledge Analysis (WEKA) [14] to generate the models and trained the data. Lastly, two datasets are constructed that are *Experiment 1* (feature selection using TF-IDF algorithm) and *Experiment 2* (feature selection using MTF-IDF algorithm). In this paper, we use similar sets of data as our main focused is to improve the feature selection approach.

3.2. Feature Selection and Extraction

Samples of Android data that have been extracted into xml file format will undergo the feature selection process. Next, the feature vector components are generated by analyzing the database. The irrelevant and redundant features will be removed in the feature selection process [15]. Then, features used are categorized into two types that are API call and dangerous permission.
The API call feature is used to extract knowledge describing behavior of Android malware. Table 1 shows the description of API call features used in this experiment. The dangerous permission feature is chosen because, through installation, each application will ask for permission to use the systems data and features [16]. Application with dangerous permission can access user's private information and affect the stored data or operation of others application. Table 2 shows list of permission based features. Both fea-

tures selected are derived from weight value calculated using TF-IDF and MTF-IDF.

Table 1: API call Based features

Type	Features	Explanation
Accounts (AccountManager)	*addOnAccounts UpdatedListener*	Notified listener whenever user or abstract account authenticator (class) made changes to accounts.
App (Activity)	*startActivity*	Launch a new activity.
Content (ContentResolver)	*Query*	Query the given URI, returning a cursor (interface) over the result set with support for cancellation.
Content (Context)	*sendBroadcast*	Broadcast the given intent to all interested broadcast receivers (class), allowing an optional required permission to be enforced.
	startActivity	Launch a new activity.
	startService	Request that a given application service be started.
Content (PackageManager)	*setComponentEnabledSetting*	Set the enabled setting for a package component (activity, receiver, service, provider).
Location (LocationManager)	*getBestProvider*	Returns the name of the provider that best meets the given criteria.
OS (PowerManager.WakeLock)	*acquire*	Acquires the wake lock.
Telephony (SmsManager)	*sendTextMessage*	Send a text based SMS using.
Webkit (WebView)	*WebView*	A View that displays web pages.
Java (net)	*HttpURLConnection*	A url connection with support for HTTP-specific features.
	Socket	An endpoint for communication between two machines.
Java (net.URL)	*openConnection*	Returns a url connection (class) instance that represents a connection to the remote object referred to by the URL.
	openStream	Opens a connection to this URL and returns an input stream for reading from that connection.

3.3. Modified Term Frequency Inverse Document Frequency (MTF-IDF) Algorithm

The proposed MTF-IDF algorithm is modified based on the TF-IDF algorithm. The TF-IDF approach gives attention on both sample and feature. However, it is unaware to the class labels in the sample, which can lead to inappropriate scaling for some features [7]. Therefore, we proposed the modified TF-IDF (MTF-IDF) to reduce the inappropriate scaling in feature selection.

This work focus on reducing inappropriate scaling by focusing features in the sample for the whole document compare with previous work [9]. Equation Eq. 6 and Eq. 7 show the MTF-IDF equation. Eq. 6 shows the MTF-IDF algorithm where, tf_{ij} is the value of Android malware feature's i in sample j. While, $\max(tf_i)$ is a highest value of Android malware feature's i in all sample. IDF in Eq. 7 represent N, the total number of Android malware feature in the dataset, where n_j is number of Android malware feature appear in sample j. Therefore, the modified weight equation is defined in Eq. 8 where W_i is weighted calculation by multiply the amount of TF and IDF.

$$TF_{ij} = \frac{tf_{ij}}{\max(tf_i)} \tag{6}$$

$$IDF_j = \log\left(\frac{N}{n_j}\right) \tag{7}$$

$$W_j = TF_{ij} \times IDF_j \tag{8}$$

Table 2: Dangerous Permission Based features

Type	Permission	Description
Calendar	*write_calendar*	Allows an application to write the user's calendar data.
Camera	*Camera*	Required to be able to access the camera device.
Contacts	*read_contacts*	Allows an application to read the user's contacts data.
	write_contacts	Allows an application to write the user's contacts data.
	get_accounts	Allows access to the list of accounts in the Accounts Service.
Location	*access_fine_location*	Allows an application to access precise location.
	access_coarse_location	Allows an application to access approximate location.
Microphone	*record_audio*	Allows an application to record audio.
Phone	*read_phone_state*	Allows read only access to phone state, including the phone number of the device, current cellular network information, the status of any ongoing calls, and a list of any PhoneAccounts registered on the device.
	call_phone	Allows an application to initiate a phone call without going through the Dialer user interface for the user to confirm the call.
	process_outgoing_calls	Allows an application to see the number being dialed during an outgoing call with the option to redirect the call to a different number or abort the call altogether.
SMS	*send_sms*	Allows an application to send SMS messages.
	receive_sms	Allows an application to receive SMS messages.
Storage	*read_external_storage*	Allows an application to read from external storage.
	write_external_storage	Allows an application to write to external storage.

4. Performance Metric

In this section, the experimental setup of Android malware detection using feature selection approach is explained. All experiments conducted undergo performance analysis to validate the approach.

4.1. Experimental Setup

A total of 1000 dataset from Drebin [10] that consist of malware and benign data has been used for the experiments. The API call based and permission based feature were tested on two experiments that are *Experiment 1* and *Experiment 2*. The *Experiment 1* and *Experiment 2* are tested using TF-IDF and MTF-IDF algorithm respectively. The comparison with the existing algorithm has been done using sensitivity testing [17]. The proposed algorithm is tested based on different number of sample, different number of feature, different types of feature and different types of machine learning algorithm.

Dataset for different number of sample consists of five different number of sample that are 200, 400, 600, 800 and 1000. Features involved in each dataset for *Experiment 1* consist of 50 API call based feature that have highest TF-IDF value. *Experiment 2* contains 50 API call based features with highest MTF-IDF algorithm value.

Meanwhile for different number of feature, dataset consists of three different set that contain 10, 50 and 100 numbers of API

based features. We selected 10, 50 and 100 numbers of features that have the highest TF-IDF and MTF-IDF value for *Experiment 1* and *Experiment 2* respectively. Both *Experiment 1* and *Experiment 2* consists of 1000 samples.

Different types of feature dataset consist of five different samples that contain 200, 400, 600, 800 and 1000 data. *Experiment 1* consists of 15 features from API call and 15 features from permission based feature that have the highest TF-IDF value. While, *Experiment 2* contains 15 features from API call and 15 features permission based feature selected based on the highest MTF-IDF algorithm value.

Lastly, different types of machine learning algorithm dataset involved in this experiment have 1000 samples and 30 features for both experiments. *Experiment 1* consists of 15 features each from API call and permission based feature that have highest TF-IDF value, while *Experiment 2* contains 15 features each from API call and permission based feature with highest MTF-IDF algorithm value. For this test, we used same features for both experiment. Then, we tested the selected features using Bagging, Decision Table and Random Forest classifier algorithm using 10 folds cross-validation.

4.2. Result and Discussion

This section shows the analysis result for Experiment 1 (TF-IDF) and Experiment 2 (MTF-IDF).

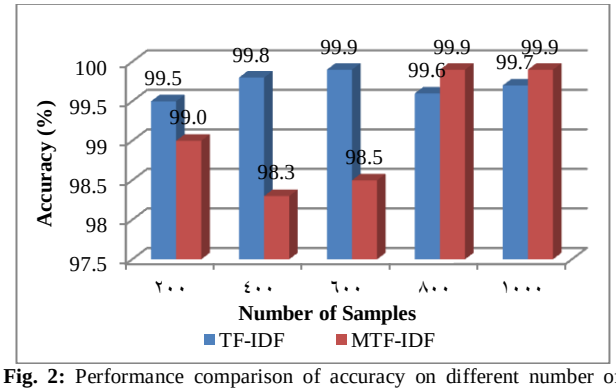

Fig. 2: Performance comparison of accuracy on different number of sample

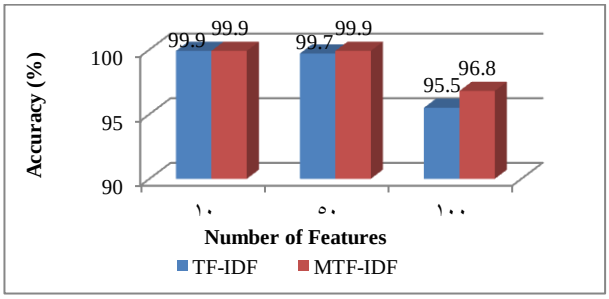

Fig. 3: Performance comparison of accuracy on different number of feature

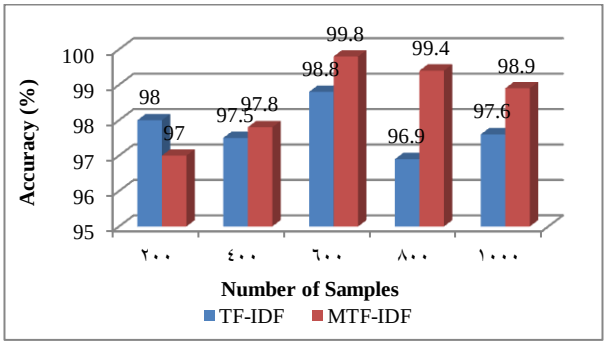

Fig. 4: Performance comparison of accuracy on combination of different types of feature

Figure 2 shows the accuracy value based on different number of samples for API call. TF-IDF achieves highest accuracy for set of dataset that contain 200, 400 and 600 samples. The MTF-IDF algorithm achieves highest percentage for dataset contain 800 and 1000 samples with 99.9% accuracy. TF-IDF shows highest accuracy when analysis small data than slightly decrease when the analyzed data increase, however MTF-IDF can maintain high accuracy with big sample data.

Figure 3 shows the accuracy value based on different number of features. MTF-IDF achieve highest accuracy for each set of dataset that contain 10, 50 and 100 features with accuracy 99.9%, 99.9% and 96.8% respectively. While the TF-IDF gain accuracy for each dataset with 99.9%, 99.7% and 95.5%.

Figure 4 shows the accuracy value based on combination of features. TF-IDF achieves highest accuracy for set of dataset that contain 200 samples with accuracy 98%. While for MTF-IDF gain highest accuracy for set of dataset that contain 400, 600, 800 and 1000 with accuracy 97.8%, 99.8%, 99.4% and 98.9% respectively.

Experiment on different types of machine learning algorithm is to test the performance of TF-IDF and MTF-IDF in terms of its accuracy. MTF-IDF shows highest accuracy as compare to TF-IDF when tested using Bagging, Decision Table and Random Forest classifier as shown in Table 3.

Table 3: Evaluation performance using different types of Machine Learning Algorithm

Algorithm	Experiment	TP Rate	FP Rate	Accuracy(%)
Bagging (meta)	TF-IDF	0.954	0.046	95.4
	MTF-IDF	0.976	0.032	97.6
Decision Table (rules)	TF-IDF	0.950	0.047	95.0
	MTF-IDF	0.968	0.040	96.8
Random Forest (tree)	TF-IDF	0.976	0.026	97.6
	MTF-IDF	0.989	0.012	98.9

5. Conclusion

In this paper, we proposed feature selection algorithm to detect Android malware. Using feature selection, only helpful feature selected during Android malware analysis. Noises features will be removed based on the weighted value. Using different types of experimental setup; such as testing based on different number of sample, different number of features, different types of feature and different types of machine learning algorithm, the proposed algorithm is tested. Both modified and existing algorithm are applied for each Android feature in sample, then only the features with higher value involve in the analysis. The analysis results show feature selection using MTF-IDF can improve malware detection analysis. MTF-IDF proved either using various kinds of feature or various kinds of dataset size, algorithm still effective for Android malware detection. In general, MTF-IDF algorithm proved that it could give appropriate scaling for all features in analyzing Android malware detection. For future research, we will consider other Android malware feature to extend the Android malware detection technique and could be applied not only for Android but also for other medium such as iOS and personal computer.

Acknowledgement

The authors express appreciation to the Universiti Tun Hussein Onn Malaysia (UTHM). This research is supported by Postgraduate Research Grant vot number U610 and Short Term Grant vot number U653.

References

[1] J. Jang, J. Yun, A. Mohaisen, J. Woo, and H. K. Kim, "Detecting

and Classifying Method Based on Similarity Matching of Android Malware Behavior with Profile," Springerplus, vol. 5, no. 1, p. 1, 2016.
[2] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond Accuracy, F-score and ROC: A Family of Discriminant Measures for Performance Evaluation," 2006.
[3] A. Feizollah, N. B. Anuar, R. Salleh, and A. W. A. Wahab, "A review on feature selection in mobile malware detection," Digital Investigation. 2015.
[4] S. Y. Yerima, S. Sezer, and I. Muttik, "High Accuracy Android Malware Detection Using Ensemble Learning.," IET Inf. Secur., 2015.
[5] A. Bhattacharya and R. T. Goswami, "Comparative Analysis of Different Feature Ranking Techniques in Data Mining-Based Android Malware Detection," in Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications : FICTA 2016, Volume 1, S. C. Satapathy, V. Bhateja, S. K. Udgata, and P. K. Pattnaik, Eds. Singapore: Springer Singapore, 2017, pp. 39–49.
[6] N. H. Mazlan and I. R. A. Hamid, "Using Weighted Based Feature Selection Technique for Android Malware Detection," in Mobile and Wireless Technologies 2017, 2018, pp. 54–64.
[7] G. Forman, "BNS Feature Scaling: An Improved Representation Over TF-IDF for SVM Text Classification," Proc. 17th ACM Conf. Inf. Knowl. Manag., 2008.
[8] A. Shabtai, R. Moskovitch, Y. Elovici, and C. Glezer, "Detection of Malicious Code by Applying Machine Learning Classifiers on Static Features: A state-of-the-art survey," Inf. Secur. Tech. Rep., vol. 14, no. 1, pp. 16–29, 2009.
[9] W. Li, J. Ge, and G. Dai, "Detecting Malware for Android Platform: An SVM-Based Approach," in Proceedings - 2nd IEEE International Conference on Cyber Security and Cloud Computing, CSCloud 2015 - IEEE International Symposium of Smart Cloud, IEEE SSC 2015, 2016.
[10] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck, "DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket.," in NDSS, 2014.
[11] L. Breiman, "Random Forests," Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001.
[12] R. Tian, "An Integrated Malware Detection and Classification System," 2011.
[13] N. Peiravian and X. Zhu, "Machine Learning for Android Malware Detection Using Permission and API Calls," in 2013 IEEE 25th International Conference on Tools with Artificial Intelligence, 2013, pp. 300–305.
[14] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: An Update," ACM SIGKDD Explor. Newsl., vol. 11, no. 1, pp. 10–18, 2009.
[15] V. M. V, P. Vinod, and D. K. A, "Heterogeneous Feature Space for Android Malware Detection," Eighth International Conference on Contemporary Computing, {IC3} 2015, Noida, India, August 20-22, 2015. pp. 383–388, 2015.
[16] Android Developers, "Permissions," https://developer.android.com/index.html, 2016. [Online]. Available: https://developer.android.com/guide/topics/permissions/index.html. [Accessed: 22-Dec-2016].
[17] I. R. A. Hamid and J. H. Abawajy, "An Approach for Profiling Phishing Activities," Comput. Secur., 2014.