Applied Information Technology And Computer Science Vol. 2 No. 2 (2021) 302-314

AITCS

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/aitcs
e-ISSN :2773-5141

Feature Selection Approach for Android Malware Detection Using Information Gain

Nur Atikah Azhari, Isredza Rahmi A Hamid*

Fakulti Sains Komputer dan Teknologi Maklumat,
Universiti Tun Hussein Onn Malaysia, Batu Pahat, 86400, Malaysia

DOI: https://doi.org/10.30880/aitcs.2021.02.02.021
Received 15 June 2021; Accepted 09 September 2021; Available online 30 November 2021

Abstract: Malware was designed to damage computer systems without user knowledge. Android is one of the platforms that usually been attacked by malicious software. In this paper, a feature selection approach for android malware detection using Information Gain is used as not all features used will give the same result. Information Gain algorithm reduce the number features and only select the best features to detect android malware. Two datasets were selected from Figshare and Malgenome. Then, these datasets are divided it into two class which are Benign and Malware. We extracted 14 features based on IG value. Then, the dataset is tested on Random Forest algorithm using WEKA tools in term of accuracy, positive rate, and false positive rate. The selected features produce promising result with 83.4% and 71% accuracy value for Malgenome and Figshare, respectively.

Keywords: Android Malware, malware detection, IG

1. Introduction

Nowadays, all information stored digitally can be accessed via the Internet at a cheaper rate. The Internet and the social media have improved significantly how society communicates. Everything is done easily and efficiently maintained. However, the Internet and social media still creates some problem regarding security for personal and confidential information of the individual such as malware [1]. This also increased the number of malware propagation channels available[2]. Malware has a tremendous impact to the world. Malicious software is software that is designed to destroy computer systems and programmed[3] which infects computer systems, deletes data files, and steals valuable information. Malware can attack personal and organization computer systems. Malware can take many forms, including virus, worm, Trojan, and spyware. Every year, malware damages a large number of computer systems[4]. Feature selection is automatically select features in the data that are most relevant for the problem. Feature selection methods can be used to identify and remove irrelevant, and redundant. Attributes from data that do not contribute to the accuracy of a predictive model. Fewer features are preferable to reduce the complexity of the model and easy to understand. However, it is ddifficult on chosen the right features to gain the highest result. Therefore, we proposed feature selection approach using Information Gain to detect android malware. The focus of feature selection is to select a subset of variables from the input which can efficiently describe the input data. Moreover, feature selection reduceed effects from noise or irrelevant variables and still provide good prediction results [5]. The IG

*Corresponding author: rahmi@uthm.edu.my

will rank the features based on its importance. Then, we extracted 14 features from the dataset. Finally, we tested the dataset using Random Forest Algorithm. The objectives of this research are as follows:

- To design feature selection approach to detect android malware.
- To develop feature selection for android malware detection model using Information Gain algorithm.
- To evaluate the proposed feature selection approach for android malware detection in terms of accuracy rate, True Positive (TP) and False Positive (FP) using Random Forest algorithm.

The rest of the paper is organized as follows: Section 2 will discuss about literature review of android malware detection. Next, section 3 will discuss about the methodology of Feature Selection Approach to detect android malware. Section 4 & Section 5 will discuss about the conclusion about the work and the future work, respectively.

2. Literature Review

This section discusses about android malware, android malware features and the existing research about the android malware detection approach.

2.1 Android Malware

The Android application is the most rapidly expanding mobile application platform. According to Lookout report[6], the Android application is growing three times than the Apple's Application Store. However, unlike Apple's Application Store, which each available application is manually scanned and checked by software security experts, there is lack of complete application inspecting process before the applications being published on the Android application Google. Android application Google employs passive mechanism that enables anyone to publish applications on the Android market[7]. When the application is reported as malware by the users, it will be removed. The accessibility of the Android market attracts both benign and malicious developers. Besides that, Android allows the installation of third-party applications, which may contribute to the spread of Android malware. There are about 130 permissions that given access. User would be asked whether to approve or reject all requested permissions by the application installed[8].

2.2 Android Malware Features

Android malware features in Table 1 are API call signature, manifest permission, and network features. An Application Programming Interface (API) is a system of regulations (called "code") and specifications that programs can use to communicate with one another. So, the Android system provides a framework API through which Applications can interact with the underlying Android system. The framework API is formed of a core set of packages and classes[9].

Every Android application package (APK) has an Android Manifest.xml file in its root directory as it is a certain permissions to access data from the other application to the android itself [6]. The Manifest.xml file includes essential information about the application to the Android system and application user. This include to get access to the restricted platform such as requests the right access to the user phone contacts database “android.permission.READ CONTACTS”[6].

Almost all behaviors of malware can be accomplished through the network interface. Network traffic has the ability to expose malicious traces of malware[10]. Examples of network traffic feature is Hypertext Transfer Protocol (HTTP). The HTTP protocol is the most common protocol for mobile applications. HTTP packets contain a lot of important information that can be used to identify and classify network traffic. Next is Transmission Control Protocol (TCP). TCP is famous transport layer protocol. The flow of TCP is a session that can identify and review as a packet set with the same 5-tuple that contain of source IP, source port, destination IP, destination port and protocol type.

Table 1: Android malware features

Features	Description
API call signature	The API calls provide means for the app to interact with the device; hence, their static inspection gives information about their run-time behavior.
Manifest permission	Request permissions to access to the sensitive data.
Network traffic	Network traffic or data traffic is the amount of data that moving across the computer or device network at given time. is divided into data packets and transmitted across a network before being reconfigured by the receiving device or computer

This work considers using manifest permission feature because android is an open-source platform which allowed user to download any application even from untrusted source. By having this permission list and description about these permission features, it may help the user to defense and detect if the application could be harmful to user. Users can decide not to install any application if the application have unnecessarily requested permission to steal user's data. Table 2 shows the description of manifest permission features with its description.

Table 2: Manifest permission[11]

Features	Description
READ_PHONE_STATE	Allows read only access to phone state, including the current cellular network information, the status of any ongoing calls
SEND_SMS	Allows an application to send SMS messages.
WRITE_APN_SETTING	Allows applications to write the apn settings and read sensitive fields of an existing apn setting like user and password.
RECEIVE_SMS	Allows an application to receive SMS messages
USE_CREDITIAL	Allows the app to request authentication tokens.
AUTENTICATE_ACCOUNTS	Allows the app to use the account authenticator capabilities of the Account Manager, including creating accounts and getting and setting their passwords.
INTERNET	Allows applications to open network sockets.
WRITE_SYNC_SETTINGS	Allows applications to write the sync settings
BLUETOOTH	Allows applications to connect to paired Bluetooth devices.
READ_HISTORY_BOOKMARKS	Allows the app to read the history of all URLs that the Browser has visited, and all of the Browser's bookmarks
READ_SYNC_SETTINGS	Allows applications to read the sync settings.
CHANGE_WIFI_STATE	Allows applications to access information about Wi-Fi networks.
RECORD_AUDIO	Allows an application to record audio.
READ_CONTACTS	Allows an application to read the user's contacts data.

2.3 Android Malware technique

2.3.1 Static Analysis

Android malware detection based on static analysis attempts to classify an app as malicious or benign by depending on features extracted from the app's apk, and source code[12]. This technique, however, may lead to inaccuracy positives because benign apps may also request risky permissions. Besides that, since Android 6.0, the permission model allows the user to grant access permission at run-time as needed, so some potentially risky permissions will never ever be granted (in fact, app developers often request permissions that are never used[12]. There are two type approach of static analysis which are:

- Signature Based-Approach.

For commercial anti-malware solutions, the signature-based approach to malware detection is the most popular approach. This technique collects conceptual behaviors and uses them to generate an antique signature. If a program's signature resembles the signature of previously identified malware, the program is classified as malware [13]. Despite the fact that the signature-based approach works very well for known types of malware. However, it is defeated in the case of new malware types that are unknown, which is the main disadvantage of this technique. Moreover, because the signature database is constrained, many malware samples remain unidentified because they are not matched with any type of malware in the database. As a result, new malware configurations must be updated as soon as they are discovered[13].

- Permission-Based Analysis.

For this approach, the permissions that the application requests are very important in order to obtain the privileges to access the application. By default, no application has access to the device's stored data and has no effect on the system's security. The permissions that are requested for resources are always saved in the file 'AndroidManifest.xml.' During the configuration of an application, user must allow the application to access all of the resources that it demands, but all of the listed privileges that are requested may not be required by that application for any reason. This approach only examines the manifest file (which contains only resource-related requests) and no other files. As a result, requests other than those for resources go unanalyzed and the result is inadequate. So this is the primary disadvantage of this approach[13].

2.3.2 Dynamic Analysis

Dynamic analysis-based techniques process includes the computation by dealing and testing of a program by executing it at real-time[13]. Aim to detect malware by capturing an app's runtime behaviour and attacking either standard or specific malware behaviors[12]. While execution the application, this analysis still can discover the malware as this is the advantages of this technique that can analyzes the behaviour of the application during the runtime and used it to get the valuable data and label it as target data. One of dynamic analysis is anomaly detection approach. In anomaly detection approach, the machine learning algorithm is being used in the technique to identify malicious behaviour in the application. The extracting features are from the set of available malwares are being used to train a framework that can be applied to an unknown malware type. Because this approach provides in-depth analysis, the tools require a large number of resources. the user needs to install the app on the users' device to evaluate whether it would be malicious or not. The limitation of this approach is that the legitimate application may also be classified as malware by this, if it invokes higher system calls[13].

2.4 Android Malware Detection Approach

Several android malware detection approaches have been proposed in recent years to combat android malware. Various feature selection approaches have been recently introduced to detect android malware. Works by [14], [15] and [16] apply machine learning algorithm in Android malware detection by implementing feature selection in order to get the better result.

Work by [14] proposed static feature using permission analysis to detect android malware. They list out some features related to permission and calculate the permission score by using permission score. To calculate the score, every string been given a weight based on the risk-level. The permission's weight is scored on a scale from 2 to 6 points corresponding to the extent of danger from moderate to very high. Every value is fixed till modified by the researchers. The data used are collection of 60 sample android while the malware was collected from a website that provides samples for research purposes. Every data gain is processed using decision tree classifier. The result shows that this combination give the accuracy value of 83%.

Work by [15] detect android malware using DREBIN. It is a lightweight method for detection android malware that infers detection patterns automatically and enables identifying malware directly

on the smartphone. It performs a broad static analysis that having as many features such as API calls and network addresses. The dataset that been used are the real Android applications and real malware. Almost 10000 applications are collected from google play store. The result shows that 94% of the malware were detected with a false-positive rate.

Work by [16] using hybrid features analysis to detect android malware. The features proposed hybrid analysis method for the detection of the Android malware that integrates the advantages of static and dynamic analysis methods. Based on this work, the hybrid features vector is extracted using a hybrid feature analysis method. In this work, researcher collect a total of 359 malicious apps and 500 benign apps as a sample but only 150 malicious apps and 150 benign apps are pick as the experimental samples and mixed them together as a training set. For the static features, all the permissions are extract from AndroidManifest.xml file in the APK package and top 10 permissions that occurred frequently are chosen. The result for static methods is running into different classifier model such as Random Forest and KNN model as different classifier algorithms will brings different detection effects. So, the highest accuracy is Random Forest with 92.07% for static methods and 94.89% for hybrid methods.

Table 3: Android malware detection approach

Work	Algorithm	Dataset	Result
Permission analysis for android malware detection[14]	Decision tree	Android application	83% accuracy
Drebin: Effective and Explainable Detection[6] Of Android Malware in Your Pocket	DREBIN	Android application	94% accuracy
Hfa-md: an efficient hybrid features Analysis based [7]	Random forest	150 malicious apps and 150 benign apps	94.89 % accuracy

This work is similar to [16] where we used the same machine learning algorithm that is Random Forest Algorithm. However, work by [16] focus on hybrid approach with the combination of permission and API calls features. While we focused on static detection technique and permission feature. We use different dataset from Figshare and Malgenome. Then, top 15 features were selected using Information Gain algorithm to rank the attributes. Random Forest algorithm is used because it is most suitable to detect malware for both datasets as random forests is not very sensitive to the parameters used to run it and it is easy to determine which parameters to use. It is also appropriate for use on datasets which having a large value of data and variables than samples [17]. Table 3 shows the comparison between existing research with our work.

3. Feature Selection Approach to Detect Android Malware

Figure 1 shows the basic system components and general processing steps for feature selection approach to detect android malware. The processing phases includes raw data, pre-processing, feature extraction and selection, 10 fold cross validation, classification algorithm, and the evaluation of the classification result. We used Random Forest algorithm as this classifier as it is a powerful knowledge representation and reasoning mechanism. Moreover, it is the simplest and most widely used classification method because of its manipulating capabilities of tokens and associated probabilities according to the user's classification decisions and empirical performance.

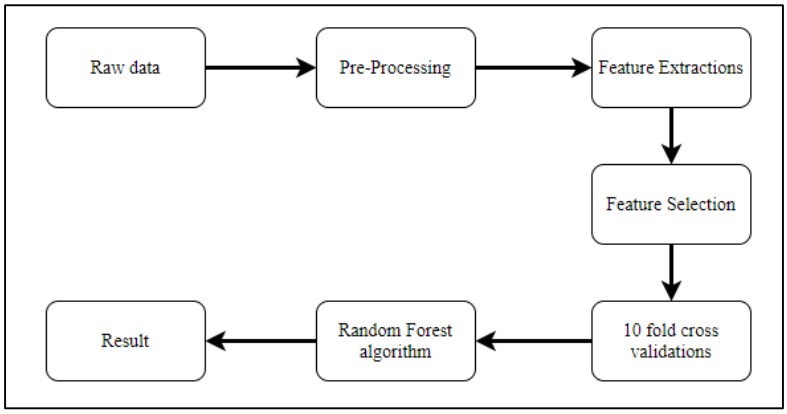

Figure 1: Feature selection to detect android malware model

3.1 Raw data

Raw data is known as unprocessed computer data. Basically, this information might be stored in a file, or a collection of numbers and characters stored in the computer's hard disk. Information that is stored in a database was often called raw data. The data can either be entered by a user or generated by the computer itself. Two types of dataset that will be used in this research are Figshare[5] and Malgenome [6]. We consider using 500 data each from Malgenome that originally have 820800 data and Figshare have 1820344.

3.2 Pre-processing

Pre-processing is a process that simply transforming raw data into understandable format[20]. It is an important process or step in data mining process which data get transformed to bring in such a state that machine can easily understand it. So, the features of the data can now be easily interpreted by the algorithm. Other than that, by having data pre-processing, it can eliminate the incorrect values because of bugs, and it can boost consistency. When having inconsistencies in data or duplicates, it will affect to the accuracy of the result. Steps involved in pre-processing are data cleaning, data integration, data transformation, data reduction and data discretization.

3.3 Feature Extraction and Feature Selection

Features extraction and selection are used separately or in combination with the aim to improve performance such as estimates accuracy, visualization knowledge[21]. Generally, features can be categorized as relevant, irrelevant, or redundant. Feature extraction is a process that identifies important features of the data. In these features, it will create the brand as new ones. It consists in transforming arbitral data into numerical features usable for the machine. The Figshare dataset consist of 14 features than 173 originally while Malgenome have 14 features than 216 before reducing. Feature selection is used for filtering irrelevant and redundant features from the dataset and it keep of the original features. After the feature selection, the list of capabilities is chosen by manual judgment or processing by algorithm. We implement feature selection approach using Information Gain (IG) algorithm. Information Gain (IG) is a feature evaluation method based on entropy that is widely used in machine learning[22]. Because it is used in feature selection, Information Gain is defined as the amount of data provided by the feature items for the document category. In order to measure the importance of features for classification, information gain is calculated by how much of a term can be used for classification of information. This is information gain formula.

$$E=-\sum_i^C P_i \log_2 p_i \qquad \textit{Eq. 1}$$

Eq. 1 is calculated for a split by subtracting the weighted entropies of each branch from the original entropy. So, when training using these metrics, the best split is chosen by maximizing Information Gain. E is representing for the entropy while p_i is the probability of randomly picking an element of

class(the proportion of the dataset made up of the class). Table 4 shows the IG value for features in Figshare dataset and Table 5 shows the IG value for features in Malgenome dataset. Based on IG value for both datasets, we select the same of 14 top features as listed in Table 6.

Table 4: IG value for features in Malgenome dataset

Features	IG value
READ_PHONE_STATE	0.1434
SEND_SMS	0.1284
WRITE_APN_SETTINGS	0.1099
RECEIVE_SMS	0.1024
USE_CREDENTIALS	0.0941
READ_SYNC_SETTINGS	0.0716
AUTHENTICATE_ACCOUNTS	0.0584
WRITE_SYNC_SETTINGS	0.0448
INTERNET	0.0439
RECORD_AUDIO	0.0401
BLUETOOTH	0.0401
READ_HISTORY_BOOKMARKS	0.0338
CHANGE_WIFI_STATE	0.0185
READ_CONTACTS	0.0109

Table 5: IG value for features in Figshare dataset

Features	IG value
FULL INTERNET ACCESS	0.1377
READ CONTACT DATA	0.0739
CHANGE WI-FI STATE	0.0331
RECORD AUDIO	0.0331
SEND SMS MESSAGES	0.0307
CREATE BLUETOOTH CONNECTIONS	0.0256
READ SYNC SETTINGS	0.0182
WRITE SYNC SETTINGS	0.0182
USE THE AUTHENTICATION CREDENTIALS OF AN ACCOUNT	0.0167
RECEIVE SMS	0.0162
READ BROWSER HISTORY	0.000
WRITE ACCESS POINT NAME SETTINGS	0.000
READ PHONE STATE AND IDENTITY	0.000
ACT AS AN ACCOUNT AUTHENTICATOR	0 .000

Table 6: Features permission used to detect android malware

Rank	Features	Description
1	READ_PHONE_STATE	Allows read only access to phone state, including the current cellular network information, the status of any ongoing calls
2	SEND_SMS	Allows an application to send SMS messages.
3	WRITE_APN_SETTING	Allows applications to write the apn settings and read sensitive fields of an existing apn setting like user and password.
4	RECEIVE_SMS	Allows an application to receive SMS messages
5	USE_CREDITIAL	Allows the app to request authentication tokens.
6	AUTENTICATE_ACCOUNTS	Allows the app to use the account authenticator capabilities of the Account Manager, including creating accounts and getting and setting their passwords.

Table 6: (cont.)

Rank	Features	Description
7	INTERNET	Allows applications to open network sockets.
8	WRITE_SYNC_SETTINGS	Allows applications to write the sync settings
9	BLUETOOTH	Allows applications to connect to paired Bluetooth devices.
10	READ_HISTORY_BOOKMARKS	Allows the app to read the history of all URLs that the browser has visited, and all of the Browser's bookmarks
11	READ_SYNC_SETTINGS	Allows applications to read the sync settings.
12	CHANGE_WIFI_STATE	Allows applications to access information about Wi-Fi networks.
13	RECORD_AUDIO	Allows an application to record audio.
14	READ_CONTACTS	Allows an application to read the user's contacts data.

3.4 N-fold cross validation

The accuracy of the machine learning algorithm can be carried out in the testing process. We consider using 10-fold cross-validation to select the most effective and best parameters for the algorithms used in this research to train and test the sample data. It is difficult to track every single data which can cause issues or problems and provide an insufficient result in the end. Thus, 10-fold Cross validation was chosen to avoid data bias as such issues and problems can be identified from multiple input features. This process helps to avoid two major problems which is overfitting and underfitting. It helps to detect the quality of the model ensuring best performance. Training dataset, it contains a known output where the training dataset allows the machine learning algorithm to perform the correctional task that will helps to learn the characteristic, identified and cluster the dataset to be generalize. Next, testing data used test the model's prediction based on the training dataset, where the output already known whose performance can be estimated and the accuracy of the machine learning algorithm can be expected. Then the data will be run using WEKA.

3.5 Classification algorithm

Random Forest algorithm is a supervised classification algorithm. Random Forest is an ensemble learning based classification and regression technique. It is one of the commonly used predictive modelling and machine learning technique[23]. This algorithm can be used for classification and regression tasks**.** The random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. One big advantage of random forest is that it can be used for both classification and regression problems, which form most current machine learning systems. By having this algorithm, it has formula that may use based on classification and regression problem. Formula for regression: using Mean Squared Error (MSE)

$$MSE= \frac{1}{N}\sum_{i=1}^{N}(fi-yi)^2 \qquad Eq.\ 2$$

Eq. 2 calculates the distance of each node from the predicted actual value, helping to decide which branch is the better decision for the forest. Here, *yi* is the value of the data point that are testing at a certain node and *fi* is the value returned by the decision tree.

3.6 Hardware and Software Specification

The hardware used are stated in the Table 7.

Table 7: Hardware Specification

Hardware	Description
Aspire ES1-432 Data Point 2	Windows edition Windows 10 Home Single Language © 2004 All right reserved

For software, we used in this project is Waikato Environment for Knowledge Analysis (WEKA). Weka is open-source software and have a tool that carries the data pro-processing, classifications, regressions, association rules and visualizations features. It is easy to carry out the software.

4. Performance Analysis

4.1 Experimental Setup

This section presents the experimental setup. In this study, the classification was performed using WEKA (Waikato Environment for Knowledge Analysis). For this experiment, we used freely available dataset; Figshare and Malgenome. We generated 2 sets of datasets which consists of 500 malware and 500 benign namely as Figshare and Malgenome. Data cleaning is performed to combine all the extracted features. Then, the Information Gain will be performed to select only the top ranked features to be used. The selected ranked attributes are illustrated in Table 6.

4.2 Performance Metric

In order to measure the effectiveness of the classification, we refer to three possible outcomes that are True Positive (TP), False Positive (FP) and Accuracy.

- True Positive (TP): is malware is detected by the classifier?
- False Positive (FP): is malware incorrectly detected by the classifier?
- Accuracy: The accuracy of a classifier is given as the percentage of total correct predictions divided by the total number of instances.

$$\text{Accuracy}=\frac{(\text{correct predictions})}{(\text{all predictions})} \qquad \textit{Eq. 3}$$

4.3 Constructing Feature Matrix

In this section, we construct the feature matrix of 14 features F_i ,i =1,….,14, i for all datasets. Note that all features are in binary value. The R_i value for each feature is summarized in Table 8.

Table 8: Feature Matrix of Malgenome dataset

Features	Description	Value
F_1	READ_PHONE_STATE	R_1 ={0,1}
F_2	SEND_SMS	R_2={0,1}
F_3	WRITE_APN_SETTINGS	R_3 ={0,1}
F_4	RECEIVE_SMS	R_4 ={0,1}
F_5	USE_CREDENTIALS	R_5 ={0,1}
F_6	READ_SYNC_SETTINGS	$R_6 = \{0,1\}$
F_7	AUTHENTICATE_ACCOUNTS	$R_7 = \{0,1\}$
F_8	WRITE_SYNC_SETTINGS	$R_8 = \{0,1\}$
F_9	INTERNET	$R_9 = \{0,1\}$
F_{10}	RECORD_AUDIO	$R_{10} = \{0,1\}$
F_{11}	BLUETOOTH	$R_{11} = \{0,1\}$
F_{12}	READ_HISTORY_BOOKMARKS	$R_{12} = \{0,1\}$
F_{13}	CHANGE_WIFI_STATE	$R_{13} = \{0,1\}$
F_{14}	READ_CONTACTS	$R_{14} = \{0,1\}$

Table 9: Feature Matrix of Figshare dataset

Features	Description	Value
F_1	Full internet access	R_1 ={0,1}
F_2	Read contact data	R_2={0,1}
F_3	Change wi-fi state	R_3 ={0,1}

Table 9: (cont.)

Features	Description	Value
F_4	Record audio	$R_4 = \{0,1\}$
F_5	Send sms messages	$R_5 = \{0,1\}$
F_6	Create bluetooth connections	$R_6 = \{0,1\}$
F_7	Read sync settings	$R_7 = \{0,1\}$
F_8	Write sync settings	$R_8 = \{0,1\}$
F_9	Use the authentication credentials of an account	$R_9 = \{0,1\}$
F_{10}	Receive sms	$R_{10} = \{0,1\}$
F_{11}	Read browser history	$R_{11} = \{0,1\}$
F_{12}	Write access point name settings	$R_{12} = \{0,1\}$
F_{13}	Read phone state and identity	$R_{13} = \{0,1\}$
F_{14}	Act as an account authenticator	$R_{14} = \{0,1\}$

Let E = {e_1 ,e_2 ,.....,$e_{|E|}$} and F = {f_1 , f_2 ,.., $f_{|F|}$ } denotes all the permission and feature vector space, respectively. So, | E | is a total permission and | F | refer to size of feature vector. Let a_{ik} a be the value of *k*th feature of *i*th permission. Therefore, the presentation of each permission is $A_i = \{a_{i1}, a_{i2}, \ldots . a_{i[e]}\}$, and each permission is $A = \{a_{ik}\}$ where i = 1,2,…,|F| and k =1,2,....,|E| . Where each permission consists of A= {READ_PHONE_STATE, SEND_SMS, WRITE_APN_SETTING, RECEIVE_SMS,USE_CREDENTIAL,AUTHENTICATE_ACCOUNTS,INTERNET,WRITE_SYNC_SETTINGS, BLUETOOTH, READ_HISTORY_BOOKMARKS, READ_SYNC_SETTINGS, CHANGE_WIFI_STATE, RECORD_AUDIO, READ_CONTACTS} and B= { Full Internet access, Read contact data, Change wi-fi state, Record audio ,Send SMS messages, Create Bluetooth connections, Read sync settings, Write sync settings, Use the authentication credentials of an account, Read SMS, Receive SMS, Read browser history, Write access point name settings}.

5. Result and Discussion

This section discusses the result of the experiment and performed some analysis for performance metric of the datasets in terms of accuracy, True Positive and False Positive.

5.1 Accuracy Result

Figure 2 shows the accuracy result between Figshare and Malgenome dataset using top 14 permission call features and ranked by Information Gain (IG) algorithm. Malgenome achieved higher accuracy result with 83.4% than Figshare with 71%. This shows that Malgenome dataset give better result detect Android Malware using selected 15 permission features.

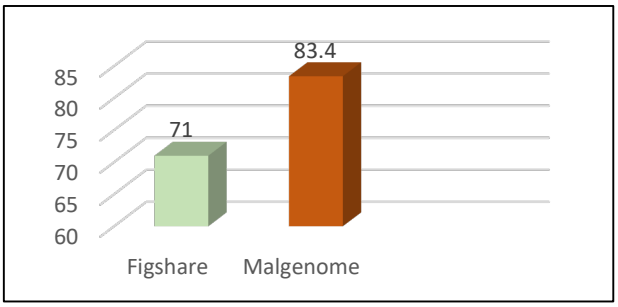

Figure 2: Accuracy value for Figshare and Malgenome

5.2 True Positive (TP) and False Positive Rate (FP)

Figure 3 shows the TP rate for Figshare and Malgenome. Figshare and Malgenome dataset achieved TP rate of 0.892 and 0.828, respectively. Moreover, Figshare achieved lower TP rate to detect benign application with 0.52 as compared to Malgenome with 0.84. This result shows that Malgenome outperformed Figshare dataset for both benign and malware data.

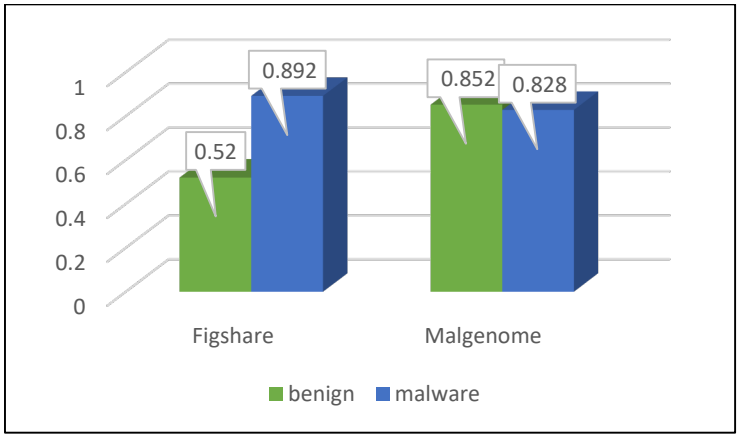

Figure 3: TP rate for Figshare and Malgenome

Figure 4 portrays the FP result for Figshare and Malgenome dataset. High FP rate means that the probability false alarm will be raised where a positive result will be given when the true value is negative. Figshare dataset shows lower FP rate as compared to Malgenome dataset for both malware and benign data. Figshare achieved FP rate with 0.108 and 0.472 for benign and malware, respectively. Malgenome obtained 0.172 for benign and 0.160 for malware. This shows that Figshare show promising result with lower FP rate than Malgenome dataset.

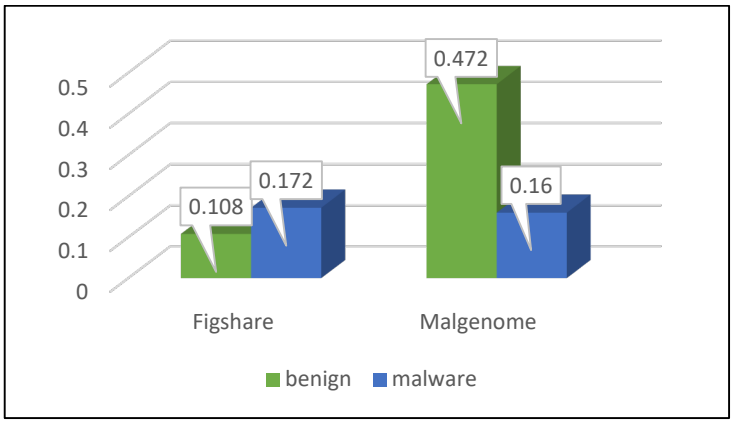

Figure 4: False Positive result for Figshare dataset and Malgenome dataset

5. Conclusion and Future Work

This work proposed feature selection approach to detect Android malware tested using Random Forest algorithm. This work used permission based features. We selected 15 features from Figshare and Malgenome. Then, we used Information Gain algorithm to select the best features for detecting android malware. The Information gain algorithm were used to rank the best features. After that, the Random Forest algorithm train and test the data using 10-fold cross validation using Waikato Environment for Knowledge Analysis (WEKA). Malgenome dataset obtained higher accuracy result with 83.4% than Figshare with 71%. This shows that the proposed feature selection approach shows promising result when tested on both datasets. As future works, we would like to investigate further on android malware detection using different features such as API call. This research could be used to assist researchers to detect android malware by using information gain of feature selection.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support and encouragement throughout the process of conducting this study.

References

[1] C. T. Dan Lo, P. Ordóñez, and C. Cepeda, "Feature selection and improving classification performance for malware detection," Proc. - 2016 IEEE Int. Conf. Big Data Cloud Comput. BDCloud 2016, Soc. Comput. Networking, Soc. 2016 Sustain. Comput. Commun. Sustain. 2016, pp. 560–566, 2016, doi: 10.1109/BDCloud-SocialCom-SustainCom.2016.87.

[2] S. K. Sahay, A. Sharma, and H. Rathore, "Evolution of Malware and Its Detection Techniques," Adv. Intell. Syst. Comput., vol. 933, no. June, pp. 139–150, 2020, doi: 10.1007/978-981-13-7166-0_14.

[3] M. A. Hama Saeed, "Malware in Computer Systems: Problems and Solutions," IJID (International J. Informatics Dev., vol. 9, no. 1, p. 1, 2020, doi: 10.14421/ijid.2020.09101.

[4] S. Mohurle and M. Patil, "A brief study of Wannacry Threat: Ransomware Attack 2017," 2017.

[5] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," Comput. Electr. Eng., vol. 40, no. 1, pp. 16–28, 2014, doi: 10.1016/j.compeleceng.2013.11.024.

[6] J. Sahs and L. Khan, "A machine learning approach to android malware detection," Proc. - 2012 Eur. Intell. Secur. Informatics Conf. EISIC 2012, pp. 141–147, 2012, doi: 10.1109/EISIC.2012.34.

[7] S. Alsoghyer and I. Almomani, "On the Effectiveness of Application Permissions for Android Ransomware Detection," Proc. - 2020 6th Conf. Data Sci. Mach. Learn. Appl. CDMA 2020, no. c, pp. 94–99, 2020, doi: 10.1109/CDMA47397.2020.00022.

[8] D. J. Wu, C. H. Mao, T. E. Wei, H. M. Lee, and K. P. Wu, "DroidMat: Android malware detection through manifest and API calls tracing," Proc. 2012 7th Asia Jt. Conf. Inf. Secur. AsiaJCIS 2012, pp. 62–69, 2012, doi: 10.1109/AsiaJCIS.2012.18.

[9] Y. Ki, E. Kim, and H. K. Kim, "A novel approach to detect malware based on API call sequence analysis," Int. J. Distrib. Sens. Networks, vol. 2015, 2015, doi: 10.1155/2015/659101.

[10] S. Wang, Z. Chen, Q. Yan, B. Yang, L. Peng, and Z. Jia, "A mobile malware detection method using behavior features in network traffic," J. Netw. Comput. Appl., vol. 133, no. January, pp. 15–25, 2019, doi: 10.1016/j.jnca.2018.12.014.

[11] U. Pehlivan, N. Baltaci, C. Acarturk, and N. Baykal, "The analysis of feature selection methods and classification algorithms in permission based Android malware detection," IEEE SSCI 2014 2014 IEEE Symp. Ser. Comput. Intell. - CICS 2014 2014 IEEE Symp. Comput. Intell. Cyber Secur. Proc., 2014, doi: 10.1109/CICYBS.2014.7013371.

[12] L. Onwuzurike, M. Almeida, E. Mariconti, J. Blackburn, G. Stringhini, and E. De Cristofaro, "A Family of Droids-Android Malware Detection via Behavioral Modeling: Static vs Dynamic Analysis," 2018 16th Annu. Conf. Privacy, Secur. Trust. PST 2018, no. Pst, 2018, doi: 10.1109/PST.2018.8514191.

[13] M. Choudhary and B. Kishore, "HAAMD: Hybrid Analysis for Android Malware Detection," 2018 Int. Conf. Comput. Commun. Informatics, ICCCI 2018, pp. 1–4, 2018.

[14] N. V. Duc, P. T. Giang, and P. M. Vi, "Permission Analysis for Android Malware," Proc. 7th VAST - AIST Work. "RESEARCH Collab. Rev. Perspect., no. November 2015, pp. 207–216, 2016.

[15] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, "Drebin: Effective and Explainable Detection of Android Malware in Your Pocket," no. February, pp. 23–26, 2014, doi: 10.14722/ndss.2014.23247.

[16] A. Ferrante, M. Malek, F. Martinelli, F. Mercaldo, and J. Milosevic, "Extinguishing ransomware - a hybrid approach to android ransomware detection," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 10723 LNCS, no. May 2017, pp. 242–258, 2018, doi: 10.1007/978-3-319-75650-9_16.

[17] H. Aydadenta and Adiwijaya, "A clustering approach for feature selection in microarray data classification using random forest," J. Inf. Process. Syst., vol. 14, no. 5, pp. 1167–1175, 2018, doi: 10.3745/JIPS.04.0087.

[18] "Android Malware and Normal permissions dataset." https://figshare.com /articles/dataset/ Android_Malware_and_Normal_permissions_dataset/8963291 (accessed Feb. 14, 2021).

[19] "Malgenome Dataset." https:// figshare.com /articles/dataset/ Android_malware_ dataset _for_machine_learning_1/5854590/1?file=10391910 (accessed Feb. 13, 2021).

[20] S. García, J. Luengo, and F. Herrera, Data Preprocessing in Data Mining, vol. 72. 2015.

[21] S. Khalid, T. Khalil, and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning," Proc. 2014 Sci. Inf. Conf. SAI 2014, no. July, pp. 372–378, 2014, doi: 10.1109/SAI.2014.6918213.

[22] S. Lei, "A feature selection method based on information gain and genetic algorithm," Proc. - 2012 Int. Conf. Comput. Sci. Electron. Eng. ICCSEE 2012, vol. 2, pp. 355–358, 2012, doi: 10.1109/ICCSEE.2012.97.

[23] B. M. Khammas, "Ransomware Detection using Random Forest Technique," ICT Express, vol. 6, no. 4, pp. 325–331, 2020, doi: 10.1016/j.icte.2020.11.001.