Efficient feature selection analysis for accuracy malware classification

Efficient feature selection analysis for accuracy malware classification

R N Romli*, M F Zolkipli, and M Z Osman

Faculty of Computer Systems & Software Engineering University Malaysia Pahang, 26200 Gambang, Malaysia

*Corresponding author**:** rahiwan@ump.edu.my

Abstract. Android is designed for mobile devices and its open-source software. The growth and popularity of android platform are high compared to another platform. Due to its glory, the number of malware has been increasing exponentially. Android system used a permission mechanism to allow users and developers to manage their access to private information, system resources, and data storage required by Android applications (apps). It became an advantage to an attacker to violent the data. This paper proposes a novel framework for Android malware detection. Our framework used three major methods for effective feature representation on malware detection and used this method to classify malware and benign. The result demonstrates that the Random forest is with 23 features is more accurate detection than the other machine learning algorithm.

1. Introduction

Nowadays, the smartphone's lifestyle is not only for communication, but it can function as a minicomputer very well. With the rapid advancement of technology in smartphones, every task can be done faster and easier. Conjunction with this development, the Android OS is chosen by company and user because of its open source and availability of its application software in the market [1]. Although it brought a lot of convenience in daily lives, Android has also become the main target of malicious program attackers and various mobile phone virus attacks.

The popularity of Android also attracts malware developers who create malicious apps to steal sensitive information and break mobile systems. Unlike other platforms, such as iOS, Android allows users to install applications from unverified sources such as third-party app stores and file-sharing websites. Today’s statistics report the average person having between 60-90 apps installed on their phone [2].

Mobile malware is the most dangerous threat that causes various security and privacy incidents such as financial damage and stealing private information. According to the [3] [4] new trend of attacking is hide and steal. The way it operates is this app will hide itself, stealing precious resources and data from mobile devices that are the passport to the digital world. The vision of these hidden apps is relatively straightforward: generate money for the developer.

The pin 2019, Kaspersky solutions repelled 975 491 360 attacks launched from online resources located worldwide and detected 24 610 126 unique malicious [5]. It is expected that a large number of mobile malware will keep developed and spread to commit various cybercrimes on mobile devices. This roughly translates to an introduction of a new malicious Android app every 10 seconds. Furthermore, in the Android market, the app update is so fast, the detection of malicious Android applications has become a big challenge, and it is an urgent need to solve a problem.

Based on [6] in total, the 1,041,336 apps in this dataset contain 235 unique permissions. The most permission-hungry apps can require many permissions from users: the single highest number of permissions required by any app was 127, although it is generally quite rare for apps to require this many. Most apps request only a handful of permissions. The average (mean) app requests five permissions. Indeed, this analysis found that nearly 100,000 apps request no permissions at all.

There are protected APIs placed between Applications and Libraries. This permission is defined in Manifest file AndroidManifest.xml, which is compulsory for shipping each android app [7]. Android uses a permission-based security model to mediate access to sensitive data, e.g., location, phone call logs, contacts, emails, or photos, and potentially dangerous device functionalities, to ensure security and privacy, e.g., Internet, GPS, and camera. The Android permission model attracts emerging malware that challenges the system to exploit vulnerabilities to perform privilege escalation attacks such as permission re-delegation attacks, confused deputy attacks, and colluding attacks. As a result, users can have sensitive data leaked or subscription fees charged without their consent, and it is related to well-known Android malware such as Trojan (FakeNeflix)and Ransomware (FakeDefender.B) [8] [9] [10].

The permission system is important in placing more emphasis on controlling how applications access sensitive devices and data store. A common technique used for detecting malware is based on the malware family, where the behavior of the family is analyzed. The weakness of malware detection family-based is the extracted feature can be approximately same. Hence, the number of feature-based detection is chosen in this study.

In this paper, we assume that dynamic analysis-based methods process obfuscated malware, and we focus on the development of a static analysis-based method to distinguish between malware and benign applications. This paper proposes a novel malware detection framework based on various static features. Our framework is flexible to add a new type of features, so it is possible to utilize dynamic features in the future. In order to present the design of the security mechanism, it will be discussing in the next section.

Recently, there are many studies regarding android permission since its inception in 2008. This section will look at the direction of previous studies in permission and malware detection based on their objective, method, and evaluation.

The study published by [11] mentions that the lack of user awareness against data protection and its relationship to permission requests will attract malicious software. Thus, the author developed a framework that consists of feature selection and classification algorithms. Four feature selection methods were separately run on the dataset, which included 3,784 android applications with different feature sets. The result obtained is a higher performance: Random Forest and J48 decision tree classification algorithms for most of the selected feature selection methods. The [12] also using the same technique to increase their accuracy in malware detection.

But it different from [13][14][15], both studies using more than one feature in malware detection. Zhao uses combination Permission and API as a feature and uses FEST as a feature-based machine learning approach for malware detection. Through FEST, a new feature selection algorithm, FrequenSel was introduced to select features by finding the difference their frequencies between malware and benign apps. The result from experiments shows that feature selection developed is more suitable to feature dataset, and a model built on the selected features is effective and reliable for malware detection.

Meanwhile, the author Kim, using various seven features such as Permission, Component, Environment, String, API, OP, and Shared Lib to defined similarity-based feature extraction method for effective feature representation on malware detection. The reason the author used many features is to reflect various characteristics of applications in various aspects. In order to evaluate performance of their framework, the author compared the accuracy of their model with that of other deep neural network models.

Hui Juan Zhu introduces the model name DroidDet in order to detect Android malware. The author proposed a low-cost and high-efficient method to extract permissions, sensitive APIs, monitoring system event and permission rate as key features, and employ the ensemble Rotation Forest (RF) to construct the model to detect an Android App is malicious or not.

Work by [16], focusing in an algorithm. The author applies the method of improved naive Bayes classification in order to get more effective in Android malware detection. In addition, malware permissions and training permissions are used as the weight in this model. In feature selection, the technique Pearson correlation is applied.

The work [17] aims to develop a novel method to extract contrasting permission patterns for comparing the differences between Android benign apps and malware based on permissions and use these differences to detect Android malware. In order to have a pattern on this study, request permission and use permission are analyzed. Then, using the different permission patterns requested by the application of the permission pattern mining stage to detect the malicious Android application, the detection accuracy can reach 94%.

Jin li [18] in introduce SIGPID, a malware detection system based on permission usage analysis to cope with the rapid increase in the number of Android malware. Instead, the author developed 3-levels of pruning by mining the permission data to identify the most significant permissions that can be effective in distinguishing between benign and malicious apps. The results in this experiment indicate that Support Vector Machine (SVM) is used as the classifier, we can achieve over 90% of precision, recall, accuracy, and F-measure.

Based on previous studies, we defined that there a two major import are feature selection and classification algorithm. In additional, the effective feature analysis process will influence the efficient and accuracy of malware detection. This relation will be discussing in next section.

2. Methods

2.1 Proposed Framework

Figure 1 shows the overall architecture of our framework. The framework conducts three major methods for detection which is Feature Extraction and Mapping method, Feature Selection method, and Classification method. These processes are explained in the next subsections. Using permission as features, the binary mapping is generated first. Then apply feature selection approaches to reduce computational cost after generating the binary vectors. Finally, the selection feature vectors are fed to the classification model for malware detection.

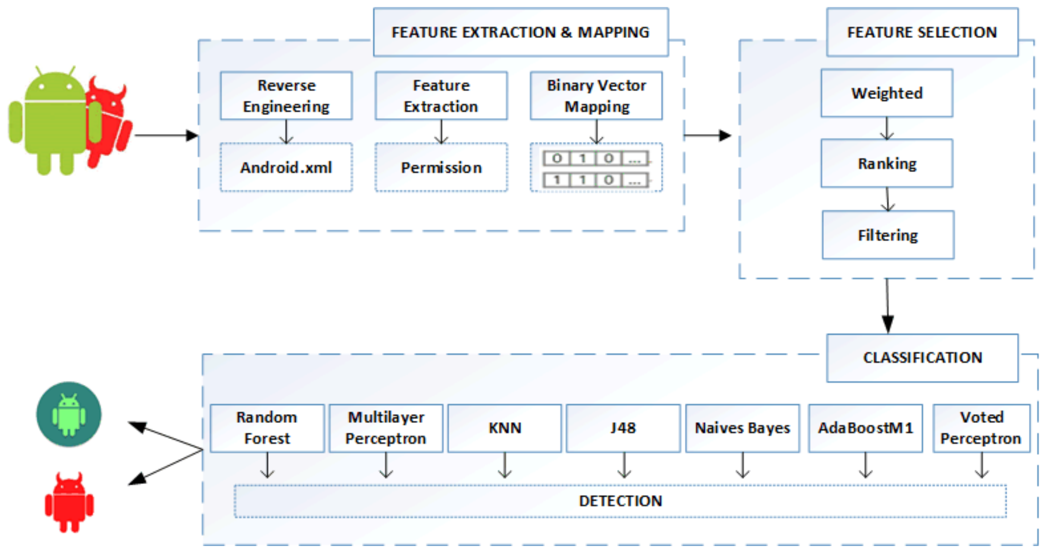

Figure 1. System Framework

2.2 Data Set
Total of 8006 android applications from Google Play Store, ContgioDump and AndroZoo are collected. From this sample, 3011 of which were found to be benign applications and 4995 malware applications. Each application was labelled as 'benign' or 'malware' to facilitate the process of supervised learning in attribute selection and classification. In order to ensure that the application is clean or contaminated by malware, the applications were uploaded into VirusTotal website to check their status. VirusTotal is an online scanner and it aggregates with many antivirus products (Sophos, Symantec, F-Secure, TrendMicro and MacAfee) to check for viruses that the user's own antivirus may have missed, or to verify against any false positives.

2.3 Feature Extraction and Mapping
This method are conducts three major processes which is reverse engineering process, feature extraction process and binary vector mapping process.

2.3.1 Reverse Engineering Process
The reverse engineering process is performed to reach the Android APK source codes. Firstly, Apk Tool is used to analyze close Android application binaries. AndroidManifest.xml is important files produces by this tool. Instead of using the batch script to retrieve permissions in AndriodManifest.xml, we mine the requested permission using Asset Packing Tools (aapt). Also, batch script allows for the running of a large number of APK file at one time.

2.3.2 Feature Extraction
The feature extraction process is conducted to obtain the essential feature data previous process. For this study, the specific permissions requested by an app in a category are compared. App that request over-privileged or uncommon permission compared to the common set of permission on that same category will indicates as malicious intention.

Permission-related information can be collected by parsing the <uses-permission> tag and the <permission> tag in the manifest file. The request permissions' names are collected from the <usespermission> tag, and the security permissions' names, permission groups and protection levels are collected from the <permission> tag. The extracted request permissions and security permissions (the tuples of name and protection level) are used as permission features.

There are four kinds of protection level in Android as discuss previously. We only extract all permissions under the "Normal" and "Dangerous" protection level. Only 62 permission features related to privacy leakage were selected to be run in the next process.

2.4 Binary Vector Mapping
Binary feature vector is needed whenever the bid size of dataset was developed. Each app in the sample was represented as a single instance with a binary vector of features and a class label indicates whether the app is benign or malicious. If the feature is present in the app it is represented by 1, if it is not present in the app, it is represented by 0.

2.5 Feature Selection
Feature Selection is the process of selecting a relevant feature for use in framework construction. In this study, filter method technique is employed to select the most relevant features and to train the different classifiers. Furthermore, not all features in permission are equally important in distinguishing the benign and malicious apps. Features are then selected based on rank to reduce the cost of running the classification algorithm on large scale dataset.

2.5.1 Weighted
Next, we implement permission weighted to perform frequency feature analysis. In this process, all the feature is given weighted based on their frequency.

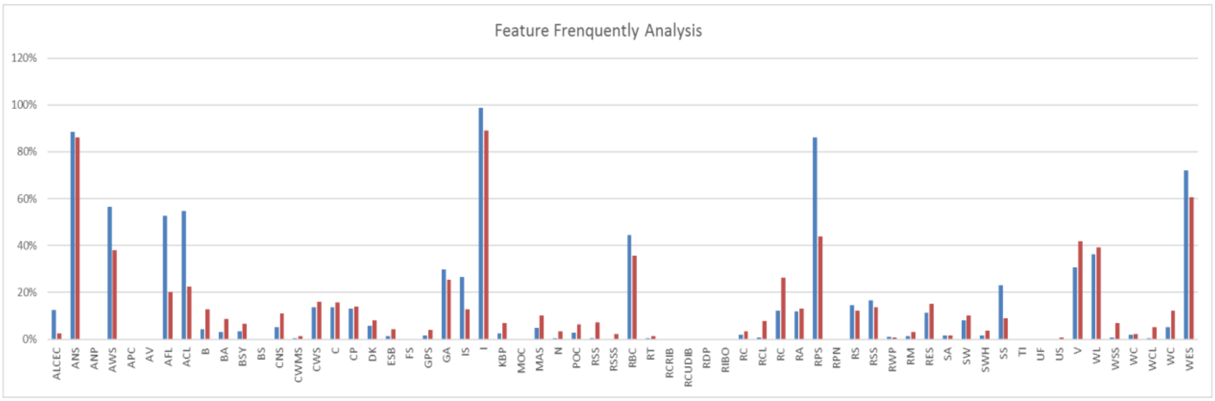

Figure 2. Popular Permissions from Android Google

Figure 2 below the occurrences of 62 popular permissions from android google. The bar graph consists from malware and benign feature. Internet permission is most popular usage for both malware and benign because this permission is allowing to open network socket. Then follow by Access Network State as second highest with 89%. The third place is Read Phone State that contribute 86%. This figure also shows that the permissions which are related to short message service (SMS), such as SEND_SMS, RECEIVE_SMS, and READ_SMS, are frequently used by malware samples. The permission like Foreground Service, Manage Own Call, Answer Phone Call are the lowest with 0% are never used by malware sample.

2.5.2 Permission Ranking

The proposed permission ranking is the process of extract the requested permissions and to identify the permissions that are unique to malware and benign apps, by calculating the contribution of each of the permission. The ranking of the permissions is based on their risk. A permission that is present only in malware samples can be termed as the one with high risk and the permissions that are requested only by benign samples can be termed as permissions with low risk.

There is some permission that is common to both malware and benign samples, such permissions need to be ignored, since they may result in ambiguity in malware detection process. This process helps to identify the risk of the permissions that are requested by the apps. Features are then selected based on rank to reduce the cost of running the classification algorithm on large scale dataset. Only 30 permissions that consider to privacy leakage were selected to be run on the several machine learning classifiers. Here, we find that 30 from 62 permission feature that frequency more that 3% above. We list these ranking from highest to low permissions in Table 1.

Table 1. Rangking of permissions

Feature	%	Feature	%	Feature	%
1. INTERNET	99	11. GET_ACCOUNTS	30	21. RECORD_AUDIO	12
2. ACCESS_NETWORK_ STATE	89	12. INSTALL_SHORTCUT		22. READ_EXTERNAL_STORAGE	11
3. READ_PHONE_STATE	86	13. SEND_SMS	23	23. SET_WALLPAPER	8
4. WRITE_EXTERNAL_	72	14. READ_SMS	17	24. DISABLE_KEYGUARD	6

STORAGE					
5. ACCESS_WIFI_STATE	57	15. RECEIVE_SMS	15	25. CHANGE_NETWORK_STATE	5
6. ACCESS_COARSE_ LOCATION	55	16. CAMERA	14	26. WRITE_CONTACTS	5
7.ACCESS_FINE_ LOCATION	53	17. CHANGE_WIFI_STATE	14 %	27. MODIFY_AUDIO_SETTINGS	5
8. RECEIVE_BOOT_ COMPLETED	45	18. CALL_PHONE	13	28. BLUETOOTH	4
9. WAKE_LOCK	36	19. ACCESS_LOCATION_EXTRA _COMMANDS	12	29. BROADCAST_STICKY	3
10. VIBRATE	31	20. READ_CONTACTS	12	30. BLUETOOTH_ADMIN	3

2.5.3 Filtering
This filtering process is important because it helps to ensure that the duplicate, used and unused permission are removed from list. Through this process, we identify that only 30 permissions are selected based on their usage and frequency used in benign and malware application. The filtered features are then sent to a machine learning process for the feature optimization which provides the number of fold values for each permission.

2.5.4 Classification
We evaluate the feature selection procedure using different classification models. For this purpose, we use six classifiers: Random Forest, KNN, J48, Naives Bayes, AdaBoostM1 and Voted Perceptron. In our experiments, we use 10-fold cross validation. Thus, the most accurate training and testing dataset distribution also the best classifier can be found.

3. Result and Discussion
In order to access the effectiveness of the framework proposed, the following evaluation metric are applied: Accuracy. Relevant confusion matrices were created from the response of classifiers where the true positive (TP) is the amount of correctly identified as malicious applications predicted as positive, false positive (FP) is amount of incorrectly identified as malicious applications as positive, true negative (TN) is the amount of correctly identified as benign applications predicted as negative, false negative (FN) is incorrectly identified as benign applications as positive.

In the experimental part, in order to avoid over fitting and perspective of stability, the 10-fold validation is applied to access the performance of framework proposed. Particularly, the samples are randomly split into ten folds, each base classifier taking one for the test set and the rest fold for the training set to construct a model.

To ensure selection of the most relevant application features for the classification stage, we only considered all features that have score value in feature ranking. We start with 30 permissions are selected through the feature selection process. To this experiment, we have used Waikato Environment for Knowledge Analysis (WEKA).

The detection performance of five (6) classifiers for Android malware detection is presented above. Each classifier performance was evaluated through five (5) performance metrics such as f-measure, recall, TPR, Precision and FPR. The table indicates that the 23 features presents a better performance compared to others number of features in 10 folder Cross Validation.

It can also be noted that Random Forest provides a good detection performance of 94.9% percent for TPR. The average TPR for each classifier was noted to be higher than 90 percent. Table above also clearly shows that the machine learning approach is effective in detecting malware such as Naiyes Bayes, Voted perceptron, K-nearest neighbors, AdaBoost and J48. Therefore, it is worth noting that feature optimization has a significant role in identifying the relevant features.

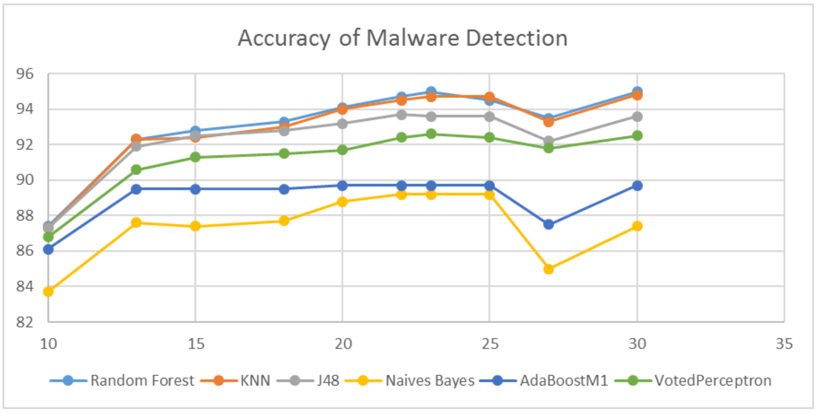

Figure 3. Accuracy of malware Detection

Figure 3 shows the accuracy of malware detection based on number of features with different type of machine learning in 10 folds. According to graph above shows that Random Forest achieved highest accuracy in 10-fold testing. The lowest accuracy when apply algorithm was achieved by Naives Nayes.

Furthermore, in an observation based in this work, the finding show that Random Forest is the best parameter at each node in decision tree is made from a randomly selected numbers in feature selection. In other word, this classifier, operates by constructing of decision trees at training time and producing the class that is mode of the classes.

All classifier shows the increasing of accuracy detection start from 18 features and the stop at number of features is 23. After that, the slope of graph continues drop until the number of features is 27. So that, the result shows that the detection of malicious apps is more effective the number of features is 23.

Tabel 2. Result Comparation with Previous Researchers

Paper	Accuracy	Dataset Size	Feature	Algorithm
[20]	91%	100,000	283	Neural Network
[21]	94.90%	3784	25	Random Forest
[22]	87.40%	9512	61	Naives Nayes
This Study	94.90%	8106	23	Random Forest

Table 2 show the comparison of result with previous researchers. This study only uses smallest features variable numbers compared to the other researchers. Even though our framework did not achieve the best accuracy, but we will add a few more feature variables to get better result.

4. Conclusion

In this paper, we have proved and proposed a framework for significant feature in malware detection on Android applications focusing on the leakage of privacy information. The study shows that Random forest classifier with 10-fold cross validation reported the highest accuracy compared to the other classifiers. Compared to previous research related to category-based, our framework able to achieved high accuracy with only small number of features with 23.

For future works, we will be focusing and consider two aspects in malware detection. First, considering adding other static feature such as API, intent, broadcast receiver or strings in training the classifier and perhaps may increase the detection accuracy and reduce the FPR and last one is we will consider to integrating with dynamic detection technique by profiling features.

Acknowledgements
This research paper and work was supported under University of Malaysia Pahang research grant RDU 1803142. The authors would like to thank you to Fakulti Komputeran, UMP and Kementerian Pelajaran Malaysia (Higher Education Department) for the scholarship and facilities in supporting this research.

References:
[1] M. O. Report 2019 *Mobile Overview Report October - December 2019* (USA: scientamobile)
[2] M. Mobile and T. Report 2019 *McAfee Mobile Threat Report Mobile Malware Continues to Increase in Complexity and Scope* (USA: McAfee)
[3] M. Malware and I. Playing 2020 *McAfee Mobile Threat Report Mobile Malware Is Playing Hide and Steal* (USA: McAfee)
[4] Kujawa A, Zamora W, Umawing J, Segura J, Tsing W, Arntz P, and Boyd C 2019 *2019 State of Malware* (Ireland: Malwarebytes)
[5] "Kaspersky Security."
[6] "Google Play Store Apps Permissions," *Pew Research Centre*. [Online]. Available: www.pewresearch.org/internet/interactives/apps-permissions/.
[7] Romli R N, Zolkipli M F, Ramli M R, and Salamat M A 2018 *Int. J. Integr. Eng.* **10** 203
[8] Bhat P and Dutta A 2019 ACM Comput. Surv. **52** 21
[9] Qamar A, Karim A, and Chang V 2019 *Futur. Gener. Comput. Syst.* **97** 887
[10] Arshad S 2016 *Android Malware Detection & Protection : A Survey*
[11] Pehlivan U, Baltaci N, Acarturk C, and Baykal N 2014 *2014 IEEE Symp. Comput. Intell. Cyber Secur.* p 1
[12] Altaher A 2016 *VAWKUM Trans. Comput. Sci.* **10** 1
[13] Zhao K, Zhang D, Su X, and Li W *Proc. - IEEE Symp. Comput. Commun* p 714
[14] Kim T, Kang B, Rho M, Sezer S, and Im E G 2018 *IEEE Trans. Inf. Forensics Secur.* **14** 773
[15] Zhu H J, You Z H, Zhu Z X, Shi W L, Chen X, and Cheng L 2018 *Neurocomputing* **272** 638
[16] Shang F, Li Y, Deng X, and He D 2018 *Cluster Comput.* **21** 955
[17] Ahmadi M, Ulyanov D, Semenov D, Trofimov M, and Giacinto G 2016 *CODASPY 2016 - Proc. 6th ACM Conf. Data Appl. Secur. Priv.*, p 183
[18] Li J, Sun L, Yan Q, Li Z, Srisa-An W, and Ye H 2018 *IEEE Trans. Ind. Informatics* **14** 3216
[19] Mewton C J and Ficek Z 2007 *J. Phys. B At. Mol. Opt. Phys.* **40** 1
[20] Arslan RS, Doğru I A, and Barişçi N 2019 *Int. J. Softw. Eng. Knowl. Eng.* **29** 43
[21] Pehlivan U, Baltaci N, Acarturk C, and Baykal N 2014 *IEEE SSCI 2014 2014 IEEE Symp. Ser. Comput. Intell. - CICS 2014 2014 IEEE Symp. Comput. Intell. Cyber Secur. Proc.*
[22] Sokolova K, Perez C, and Lemercier M 2017 *Decis. Support Syst.* **93** 62