Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

# Android Malware Detection using Permission Based Static Analysis

Noor Afiza Mohd Ariffin[1,*], Hanna Pungo Casinto[1]

[1] Faculty of Computer Science and Information Technology, University of Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia

**ARTICLE INFO**

***Article history:***
Received 5 July 2023
Received in revised form 18 October 2023
Accepted 27 October 2023
Available online 12 November 2023

***Keywords:***
Android; malware detection; static analysis; permission weight

**ABSTRACT**

The increase of mobile device enhancement grows. With this development, mobile phones are supporting many programs, and everyone takes advantage of them. Nevertheless, malware applications are increasing more and more so that people can come across lots of problems. Android is a mobile operating system that is the most used on smart mobile phones. Because it is the most used and open source, it has been the target of attackers. Android security is related to the permissions allowed by users to the applications. There have been many studies on permission-based Android malware detection. In this study, a permission-based Android malware system is analyzed. Unlike other studies, we propose a permission weight approach. Each of the permissions is given a different score using this approach. Then, K-nearest Neighbor (KNN) and Naïve Bayes (NB) algorithms are applied, and the proposed method is compared with the previous studies and the expected experimental results of the proposed approach will be higher.

## 1. Introduction

Android is the leading operating system for smartphone users. According to Global Stats in 2019 counter report, there is 74.85% Android operating system used in the smartphone market and followed by iOS which is 22.94%. This mobile device operating system is now leading among other mobile operating systems. These market share improvements make the Android operating system a clear target for malicious people. According to Nokia's Threat Intelligence Report in 2018, there were 78% of malware detection events in communication service provider networks in 2018. For this reason, the development of security measures for the Android operating system is very important. High connectivity possibilities, sensor capabilities, and wireless communication facilities expose them to a wide variety of attacks. On the other hand, the limited system resources of mobile devices present new challenges in malware detection. Unlike other competing smart-mobile device platforms, such as iOS, Android allows users to install applications from unverified sources such as third-party app stores and file-sharing websites. The malware infection issue has been very serious and a recent report indicates that 97% of all mobile malware target Android devices. In 2017 alone, over 3.25 million new malicious Android apps have been uncovered. This roughly translates to an

[*] *Corresponding author.*
*E-mail address: noorafiza@upm.edu.my*

https://doi.org/10.37934/araset.33.3.8697

introduction of a new malicious Android app every 10 seconds. These malicious apps are created to perform different types of attacks in the form of trojans, worms, exploits, and viruses. Based on the Symantec report in 2017. Some notorious malicious apps have more than 50 variants, which makes it extremely challenging to detect them all.

In the existing research, malicious software detection methods use static and dynamic approaches to obtain application data [3]. The static approach aims to determine malware before the run time. It analyses the source files and classifies them according to the desired permissions and system specifications. The main advantages of this method are that it is fast, has low system resource requirements, and blocks malware without harming the system yet [3]. The weak point of the method is that it is only resistant to known methods and cannot prevent zero-day attacks.

## 2. Literature Review

The Android operating system is widely used with more than a billion of users including all kind of services (cell phone, smart phone, TV etc). According to Urcuqui-López *et al.*, [20], The amount of sensitive data “using” these technologies has increased the cyber criminal’s interest to develop tools and techniques to acquire that information or to disrupt the device’s smooth operation.

In the study of Utku et al., [21], they proposed a malware detection system which based on multilayer perceptron for detection of Android malware. They using dataset consisting of 7210 applications including malicious applications in Drebin dataset and normal applications obtained through the Google Play Store.

Azmoodeh *et al.,* [6] gave a system that calculates the risk level of applications at the time of installation. They used two parameters to calculate the risk level of the application. The first of these is the permissions required by the application. The second is the download rates and user ratings on the application market. They constructed a risk analysis of the application with a metric obtained through these two parameters. Ali *et al.,* [7] have proposed a system called 'Android Application Analyzer' that classifies installed applications as malicious or not.

The system classifies applications according to the permissions they request and offers the option of completely removing from the system for malicious applications. The system uses the Naïve Bayes algorithm for classification. Tarute *et al.,* [8] worked on a total of 558 Android Application Package (APK) files that they obtained from different data sets. A data set consisting of 330 features was obtained by writing 1 for the permissions granted by the application and 0 for any other cases. They divided the data set by 71% train and 29% test and applied the classification process. By using 6 different classifiers in the classification phase, they achieved classification success by 90% with NB, 93% with Bagging, 94% with KNN, 94% with support vector machine, 92% with stochastic gradient descent and 94% with decision tree.

Despite all the investigations that have been conducted, fewer studies have given focus to discussing root exploit malware, particularly for Android mobile devices, except for Droidanalyzer [27] and Droidexec [28]. This justifies the investigation conducted on root exploits is rare, especially involving Android. Unlike Droidanalyzer and DroidExec, this paper aims to adopt machine learning as a means to detect it. The Droidanalyzer [27] used an algorithm to calculate the MD5 hash value which was cross-referenced in the database of signatures. In comparison, the similarity recognition applied by Droidexec [28] used a structural graph constructor (i.e., function–relation graph extraction and opcode component graph constructor). Both had also overlooked detecting root exploits with the machine learning strategy.

Previous study presented a four-layer filtering mechanism, an integrated static analysis method which includes the application message summarization value (MD5), then a combination of malicious permissions, the dangerous intentions, and dangerous permissions [19]. The main objective of the research is to use an intuitive threat-degree model that detects dangerous permissions. They have achieved 100% success rate in detection through MD5 and combination permission, while 99% success rate in dangerous intention and permissions using 4006 malicious samples in their testing experiment. Thus, this method gets a higher accuracy rate using many samples and had experimented using the real mobile device. Table 1 below summarized the existing research that are related to this research area in term of methodology and their contribution.

**Table 1**
Literature review summary

| Method | Author name & (year) | Methodology | Results/ contribution |
|---|---|---|---|
| Static | Şahin *et al.,* [18] | They used a requested permission with Relevance Frequency as weighing method to the permission, they analyzed 399 samples from dataset. The analysis was conducted with 10 fold cross-validation using KNN and NB. | Their contribution to malware detection method is relatively accurate with the use of RF weighting model. They have obtained 93% accuracy in KNN while in 91% with NB. |
| Static | Song *et al.,* [19] | This integrated static model used intuitive threat-degree that detects dangerous permissions. The model is composed of four layer filtering mechanism (MD5, malicious permission, dangerous permission, dangerous intentions). This study used a dataset of 4000 malicious samples. | This model had an excellent result which achieved 100% detection rate for MD5 and combination permissions while 99% detection rate for dangerous permission and dangerous intentions. |
| Static | López and Cadavid [2] | Using a binary weighting method on the permission to detect the malware applications, employing 558 APKs file, they used 330 features from the obtained dataset. They have used 6 multiple classifiers (KNN, SVM, DT, NB, Bagging, and stochastic gradient descent). | The highest rate obtained from the 6 classifiers are KNN, SVM and Decision Tree which achieved 94% classification rate. |
| Static | Wang *et al.,* [23] | Proposed a framework to efficiently categorize malware from benign apps in a large app market. They used ensemble classifiers (KNN, SVM, NB, and Random Forest) for extracting 11 static features and employ a dataset of 100,000 benign set and 8,000 malicious app set. | Their method obtained 82% correctness in classification rate for benign and 99% rate for malware application. |
| Static | Matsudo *et al.,* [16] | They created a risk analysis system for the apps through, first, permissions that required by the app when user installed and second, from user ratings on the apps market. The aim is to measure the time and precision of the app's permissions. | This study managed to get the user to be able to distinguished malware app quickly. The system is also built for non-tech savvy to be able to detect the malapps or non-malapps they are about to install. |
| Static | Almin and Chatterjee [3] | Proposed a system called 'Android Application Analyzer' that classifies installed applications if malicious or benign. Used NB in classifying and its clustering technique to classify the permission they requested. | The system allows user to detect the malware in their mobile phones and they have given an option to remove it. |
| Static | Yang and Wen, [25] | Used static analysis for Android app with feature engineering. They used a combined features from classes.dex which is an executable file and also from Android manifest files and use it for classification during testing/training. | This method managed to get 98% accuracy rate using Random Forest classifier. |
| Static | Milosevic *et al.,* [17] | Used static analysis process through source code analysis and multiple machine learning to detect malware app. | Obtained 87.9% accuracy rate using SVM and 95% accuracy rate for the combination of code analysis machine learning algorithms. |
| Static | Wang *et al.,* [22] | Used DroidEnsemble to detect malware on Android apps through structural feature and string features. Used ensembles (KNN, SVM, and Random Forest) machine learning. Employed 1296 malapps and 1386 benign apps to evaluate the performance of the method. | It obtained 96% on string features alone and 91% with structural features. Overall, DroidEnsemble obtained 98% detection accuracy with ensemble for both features. |

| | | | |
|---|---|---|---|
| Dynamic | Feizollah *et al.,* [5] | Used AndroDialysis to evaluate the Android Intents effectiveness as a unique feature for detecting malicious applications. They employed 5,560 malapps and 1,846 benign apps to evaluate the model. | AndroDialysis obtained the detection rate of 91% accuracy. |
| Dynamic | Ahmad Firdaus and Zainal Abidin [1] | He implemented machine learning as a means to detect root exploit malware through identification of the system prototype. | Achieved accuracy rate of 95% and for the performance results, the systems obtained the value of 0.627 MB application size within in 18 seconds only. |
| Dynamic | Zaki *et al.,* [26] | Used n-gram system call sequence feature to detect malware and benign apps. Several filter and wrapper feature selection methods are also analyzed and evaluated. | This results in the increase of False Positive Rate (FPR), True Positive Rate (TPR), as well as the Linear-SVM classifier Accuracy in the classification of malicious and benign mobile malware application. |
| Dynamic | Shankar *et al.,* [11] | Used Dynamic Taint Analysis (AndroTaint) method to analyze Android malapps. This method works without modifying Android platform by using automatic tagging. Used 10-fold cross validation to measure the effectiveness of the model. | AndroTaint obtained 90% accuracy rate and the output resulted in less false negative and false positive rate. |
| Dynamic | Mahindru and Singh [14] | Used 123 extracted dynamic permissions from large dataset of 11000 apps. To evaluate the model, they used multiple machine learning classifications such as SL, DT, K-star, RF and NB. | The model obtained 99% accuracy rate from SL from among all the employed classifiers. |
| Hybrid | Arshad *et al.,* [4] | Proposed a model called SAMADroid, which aims to thoroughly investigate the detection method. It is a 3-layer hybrid detection system which are, Static and Dynamic Analysis; Machine Learning Algorithms; and Local and Remote Host. | The result of the experiment demonstrates that if the efficiency in power and storage consumption is ensured, SAMADroid obtained high accuracy in malware detection. |
| Hybrid | Kabakus *et al.,* [9] | Proposed Malicious Application Detector for Android or known as *mad4a* to detect the Android apps characteristics. This is based on analyzing the permissions and network log of applications. | Based on findings from this method, malapps tend to disable mobile data connection which is one way to detect it. This method also helped informed digital investigators of those miscalculated Android malware characteristics. |
| Hybrid | Lindorfer *et al.,* [13] | Proposed ANDRUBIS analysis system that uses dynamic taint tracking to detect malapps. It focused on identifying the nature of an application at the Dalvik VM as well as system level. | The system offers investigators a platform based on the static features and dynamic behavior of the apps to build post-processing methods. |
| Hybrid | Kuo *et al.,* [10] | Used two methods, the characteristics of the permission for static analysis, and Application Program Interface (API) from dynamic analysis. Used SVM and Random Forest machine learning in classification technique. | The highest result of the proposed model is 89% accuracy rate and 88% True Positive rate correspondingly. |
| Hybrid | Martín *et al.,* [15] | Proposed AndroPyTool, an automated framework which used the large OmniDroid dataset. 22,000 features extracted, benign and malware apps samples. | The results of this experiment show the potential usability and feasibility of the automated framework. They have also offered dataset which is publicly available. |

## 3. Methodology
### *3.1 Research Framework*

By adapting Relevance Frequency as a weighting method, KNN and Naïve Bayes classification algorithms are used to evaluate the accuracy of the proposed method as shown in Figure 1.

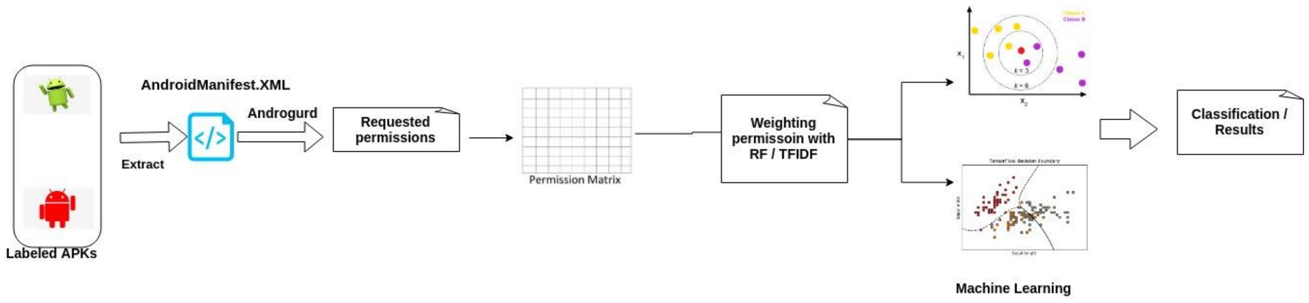

**Fig. 1.** Proposed framework

### *3.2 Performance Measure*

A Confusion matrix (refer to Table 2) is used as a technique for summarizing the classification algorithm performance. It shows the results summary of prediction on a classification problem. The key to a confusion matrix is the summarized number of correct and incorrect predictions from the actual classification with count values which are broken down by each class. It shows the insight of the errors being made by your classifier as well as the types of errors that have been made.

i. True Positive (TP) is the actual labelled positive sample in the dataset and classified as positive in the classification result.
ii. True Negative (TN) is the negative labelled sample in the dataset and is actually classified as negative in the classification result.
iii. False Positive (FP) is a sample that is actually labelled negative and classified as positive.
iv. False Negative (FN) is a sample that is actually labelled positive and classified as negative.

In this study, Accuracy and F1-measure will be used to measure the performance of the model. As Accuracy is not the only measurement for the model's accuracy, F1-score is added as it is the most commonly used performance measure in the classification method to balance the precision and recall. Here, Precision ($\pi$), recall ($\rho$), F1-score and Accuracy equation are respectively explained.

i. Precision ($\pi$): When the model is accurate/precise from those predicted positive and how many are actual positive from them. For example, in malware detection, it is the number of applications that are correctly classified as malware. Precision uses false positive rate calculation as shown in Eq. (1).

$$\pi = \frac{TP}{TP+FP} \tag{1}$$

ii. Recall ($\rho$): It calculates the number of Actual Positives the model had captured by labelling it Positive (True Positive). Thus, Eq. (2) focused on the false negative rate.

$$\rho = \frac{TP}{TP+FN} \tag{2}$$

iii. F1-measure (*F*): Eq. (3) represents the harmonic mean of precision and recall, to balance the two equations. It means to reduce both False Positives and False Negatives.

$$F1 = \frac{2\pi\rho}{\pi+\rho} \tag{3}$$

iv. Accuracy: It is used for calculating the accuracy of the model. In this case, the accuracy of the system model that detects malware applications is shown in Eq. (4).

$$Acc = \frac{TP+TN}{TP+FP+TN+FN} \tag{4}$$

**Table 2**
Confusion matrix

| | Actual Positive | Actual Negative |
|---|---|---|
| Predicted Positive | TP | FP |
| Predicted Negative | FN | TN |

### *3.3 Used Dataset*

The dataset used in this study occurred by Sahin *et al.*, [18] which contained a set of both malicious and benign software. It is composed of 200 benign applications and 199 malware apps which are originally taken from www.kaggle.com. There are 330 permissions given as features in the dataset. Thus, it would be marked as 1 if the permission is found in the application, else, the score is marked as 0. Through this process, it obtained the binary vector. The application of the weighing method on the dataset is also emphasized.

### *3.4 Static Analysis*

The static approach aims to identify malware applications before their run-time. It is used to analyze the file sources and then classify them accordingly to the system specifications and permissions. The static method is fast and has a low resource requirement for the system and it also blocks the malware without leaving any harm in the system Sahin *et al.*, [18] and Yan and Yan [24]. Technically, the main use of static analysis tools is to analyze Android APK files to inspect their various components. The following sub-sections discuss several static analysis tools.

## **4. Result and Discussion**

This section will also evaluate the proposed framework in terms of its accuracy and will be compared to the study conducted by Sahin *et al.*, [18]. Furthermore, for requested permission, the approach used to map the result to the requested permission's feature matrix is based on 22 significant permission that was introduced by Li *et al.*, [12].

The results for KNN, NB and SVM classification algorithms were tested using the 399 samples of the dataset. As previously discussed, the dataset is randomly selected and divided into 70% training data and 30% testing data. To ensure the robustness of the machine learning algorithms, the test was conducted by running 10-fold cross-validation in the experiment. The results are presented accordingly with the average score of the accuracy of every machine learning classification.

The experiment tested the performance of every classification model of machine learning after applying the weighted score. The first experiment is KNN, followed by NB, and finally the additional machine learning to this study, the SVM.

The accuracy results of every machine learning are explained in the following sections and compared with the previous study which was conducted by Sahin *et al.*, [18].

### *4.1 KNN Algorithm Classification Results*

According to the K-Nearest Neighbor classification algorithm, Sahin *et al.*, [18] obtained average results with 10 cross-validation is 93% accuracy. The result in Table 3 for the proposed method shows slightly higher than Sahin *et al.*, [18] which is 94% Accuracy. Thus, based on the proposed method using the KNN algorithm, there is a 1% average improvement in the model Accuracy. Moreover, given the results of 10-fold cross-validation, the most successful result that obtained 99% is in the eighth cross-correlation while Sahin's is in the 5-fold which gives the score of 97%.

Hence, the proposed method using KNN gets a moderately higher accuracy rate in the classification of malware and benign applications.

**Table 3**
KNN classification results using RF method

| 10-Fold CV | Şahin*, et al.,* [18] | Proposed framework |
|---|---|---|
| CV1 | 92% | 93% |
| CV2 | 92% | 90% |
| CV3 | 93% | 95% |
| CV4 | 93% | 95% |
| CV5 | 97% | 95% |
| CV6 | 92% | 90% |
| CV7 | 93% | 94% |
| CV8 | 94% | 99% |
| CV9 | 93% | 94% |
| CV10 | 95% | 95% |
| Average (accuracy) | 93% | 94% |

### *4.2 NB Algorithm Classification Results*

In the average result of Sahin *et al.*,[18], the Naïve Bayes classification algorithm obtained 90% accuracy with 10 cross-validation. The result presented in Table 4 shows no significant difference in accuracy with an average score of 90% from the proposed framework. Thus, based on the proposed method using the NB algorithm, it shows the same rate score as the classification method. On the other hand, with the results given to the 10-fold cross-validation, the most successful result that obtained 99% is in the 9th cross-correlation while Sahin's obtained the highest rate score of 92% in multiple folds.

It turns out that the NB classifier offers the same performance based on the average score but significantly higher in certain folds of cross-validation.

**Table 4**
NB classification results using RF method

| 10-Fold CV | *Şahin, et al., [18]* | Proposed framework |
|---|---|---|
| CV1 | 90% | 87% |
| CV2 | 87% | 95% |
| CV3 | 91% | 90% |
| CV4 | 89% | 95% |
| CV5 | 92% | 89% |
| CV6 | 92% | 80% |
| CV7 | 86% | 89% |
| CV8 | 92% | 85% |
| CV9 | 91% | 99% |
| CV10 | 92% | 87% |
| Average (accuracy) | 90% | 90% |

## *4.3 SVM Algorithm Classification Results*

The main contribution of this study is to employ the third machine learning algorithm which is the SVM. Table 5 shows all the results of the machine learning used in the proposed model. One of the highlights of these results is the average score of the SVM Accuracy rate. SVM obtained the highest average score of 95% in comparison to the other machine learning algorithms.

**Table 5**
SVM classification results using RF method

| | Şahin, *et al.,* [18] | | Proposed framework | | |
|---|---|---|---|---|---|
| 10-Fold CV | KNN | NB | KNN | NB | SVM |
| CV1 | 92% | 90% | 86% | 87% | 93% |
| CV2 | 92% | 87% | 90% | 95% | 95% |
| CV3 | 93% | 91% | 95% | 90% | 95% |
| CV4 | 93% | 89% | 95% | 95% | 96% |
| CV5 | 97% | 92% | 95% | 89% | 93% |
| CV6 | 92% | 92% | 90% | 80% | 93% |
| CV7 | 93% | 86% | 90% | 89% | 93% |
| CV8 | 94% | 92% | 99% | 85% | 93% |
| CV9 | 93% | 91% | 93% | 99% | 99% |
| CV10 | 95% | 92% | 95% | 87% | 97% |
| Average (accuracy) | 93% | 90% | 93% | 90% | 95% |

Support Vector Machine demonstrate a higher performance in the classification method out of the three machine learning presented. In Sahin *et al*., [18] study, they presented only two machine learning, KNN and NB that were discussed in the previous sections. Thus, the proposed framework uses another machine learning to get more accuracy in detecting and classifying malware applications.

As mentioned in the previous section, F1-score is a combination of precision and recall. It is also called the harmonious mean of both precision and recall. In a nutshell, the Precision can be considered as a measure of the exactness of the classifiers. Thus, low precision could mean a large number of False Positives. On the other hand, Recall is the measurement of a classifier's completeness and low recall means a great number of False Negatives Tait *et al*., [29]. Therefore, F1-score balanced both precision and recall, which gives more accuracy to the performance metrics.

The result on accuracy of the proposed model is explained on the above table which clearly indicated the higher performance rate on the accuracy of SVM. Below in Table 6 shows the F1-score of the proposed framework which obtained a 95% rate for KNN, 90% for NB and 94% for SVM. In comparison with Sahin *et al*., [18], the proposed framework gets a higher F1-score of 2% on KNN but obtained a lower score on NB. On the other hand, the added machine learning in this study which is SVM obtained a 94% F1-score. It has achieved at least a 1% higher rate than the other machine learning classifiers used by Sahin *et al*., [18].

**Table 6**
F1-Score comparison with the proposed framework

| | Şahin, *et al.,* [18] | | Proposed framework | | |
|---|---|---|---|---|---|
| | KNN | NB | KNN | NB | SVM |
| F1-score | 93% | 93% | 95% | 90% | 94% |

## 5. Conclusion

The importance of mobile device security development has increased throughout the years. This has been applied to smartphone and tablet users. Since the Android operating system is widely used for storing sensitive data, this has become the main target of the attackers using malapps. For this reason, a framework in this study is proposed for mobile malware detection. For each permission, we applied a weighted score to avoid a sparse matrix problem for the classification of the malware and benign apps. Thus, from the results, it has been found that the weighting method offers good results in Android malware detection. Using the three machine learning algorithms to classify the applications from malware to benign applications. SVM gets a higher rate with a score of 97% accuracy compared to other machine learning methods. It was followed by the KNN which obtained 95% indicating a slightly higher accuracy in comparison to the previous studies. In addition, KNN gets a higher score of 95% which gives a 2% improvement for the F1-score. For future studies, a large number of datasets can be tested using the proposed framework to test the model’s accuracy as well as using different machine learning algorithms.

**Acknowledgement**
This research was funded by a grant from Universiti Putra Malaysia (IPM Grant 9679500).

## References

[1] Ahmad Firdaus, Zainal Abidin. "Mobile malware anomaly-based detection systems using static analysis features/Ahmad Firdaus Zainal Abidin." PhD diss., University of Malaya, 2017.
[2] Lopez, Christian Camilo Urcuqui, and Andres Navarro Cadavid. "Machine learning classifiers for android malware analysis." In *2016 IEEE Colombian Conference on Communications and Computing (COLCOM)*, pp. 1-6. IEEE, 2016. https://doi.org/10.1109/ColComCon.2016.7516385
[3] Almin, Shaikh Bushra, and Madhumita Chatterjee. "A novel approach to detect android malware." *Procedia Computer Science* 45 (2015): 407-417. https://doi.org/10.1016/j.procs.2015.03.170
[4] Arshad, Saba, Munam A. Shah, Abdul Wahid, Amjad Mehmood, Houbing Song, and Hongnian Yu. "SAMADroid: a novel 3-level hybrid malware detection model for android operating system." *IEEE Access* 6 (2018): 4321-4339. https://doi.org/10.1109/ACCESS.2018.2792941
[5] Feizollah, Ali, Nor Badrul Anuar, Rosli Salleh, Guillermo Suarez-Tangil, and Steven Furnell. "Androdialysis: Analysis of android intent effectiveness in malware detection." *computers & security* 65 (2017): 121-134. https://doi.org/10.1016/j.cose.2016.11.007
[6] Azmoodeh, Amin, Ali Dehghantanha, and Kim-Kwang Raymond Choo. "Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning." *IEEE transactions on sustainable computing* 4, no. 1 (2018): 88-95. https://doi.org/10.1109/TSUSC.2018.2809665
[7] Ali, Feizollah. "A malware analysis and detection system for mobile devices/Ali Feizollah." PhD diss., University of Malaya, 2017.

[8] Tarute, Asta, Shahrokh Nikou, and Rimantas Gatautis. "Mobile application driven consumer engagement." *Telematics and Informatics* 34, no. 4 (2017): 145-156. https://doi.org/10.1016/j.tele.2017.01.006
[9] Kabakus, Abdullah Talha, and Ibrahim Alper Dogru. "An in-depth analysis of Android malware using hybrid techniques." *Digital Investigation* 24 (2018): 25-33. https://doi.org/10.1016/j.diin.2018.01.001
[10] Kuo, Wen-Chung, Tsung-Ping Liu, and Chun-Cheng Wang. "Study on android hybrid malware detection based on machine learning." In *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*, pp. 31-35. IEEE, 2019. https://doi.org/10.1109/CCOMS.2019.8821665
[11] Shankar, Venkatesh Gauri, Gaurav Somani, Manoj Singh Gaur, Vijay Laxmi, and Mauro Conti. "AndroTaint: An efficient android malware detection framework using dynamic taint analysis." *2017 ISEA Asia security and privacy (ISEASP)* (2017): 1-13. https://doi.org/10.1109/ISEASP.2017.7976989
[12] Li, Jin, Lichao Sun, Qiben Yan, Zhiqiang Li, Witawas Srisa-An, and Heng Ye. "Significant permission identification for machine-learning-based android malware detection." *IEEE Transactions on Industrial Informatics* 14, no. 7 (2018): 3216-3225. https://doi.org/10.1109/TII.2017.2789219
[13] Lindorfer, Martina, Matthias Neugschwandtner, Lukas Weichselbaum, Yanick Fratantonio, Victor Van Der Veen, and Christian Platzer. "Andrubis--1,000,000 apps later: A view on current Android malware behaviors." In *2014 third international workshop on building analysis datasets and gathering experience returns for security (BADGERS)*, pp. 3-17. IEEE, 2014. https://doi.org/10.1109/BADGERS.2014.7
[14] Mahindru, Arvind, and Paramvir Singh. "Dynamic permissions based android malware detection using machine learning techniques." In *Proceedings of the 10th innovations in software engineering conference*, pp. 202-210. 2017. https://doi.org/10.1145/3021460.3021485
[15] Martín, Alejandro, Raúl Lara-Cabrera, and David Camacho. "Android malware detection through hybrid features fusion and ensemble classifiers: The AndroPyTool framework and the OmniDroid dataset." *Information Fusion* 52 (2019): 128-142. https://doi.org/10.1016/j.inffus.2018.12.006
[16] Matsudo, Takayuki, Eiichiro Kodama, Jiahong Wang, and Toyoo Takata. "A proposal of security advisory system at the time of the installation of applications on Android OS." In *2012 15th international conference on network-based information systems*, pp. 261-267. IEEE, 2012. https://doi.org/10.1109/NBiS.2012.110
[17] Milosevic, Nikola, Ali Dehghantanha, and Kim-Kwang Raymond Choo. "Machine learning aided Android malware classification." *Computers & Electrical Engineering* 61 (2017): 266-274. https://doi.org/10.1016/j.compeleceng.2017.02.013
[18] Şahın, Durmuş Özkan, Oğuz Emre Kural, Sedat Akleylek, and Erdal Kiliç. "New results on permission based static analysis for Android malware." In *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, pp. 1-4. IEEE, 2018. https://doi.org/10.1109/ISDFS.2018.8355377
[19] Song, Jun, Chunling Han, Kaixin Wang, Jian Zhao, Rajiv Ranjan, and Lizhe Wang. "An integrated static detection and analysis framework for android." *Pervasive and Mobile Computing* 32 (2016): 15-25. https://doi.org/10.1016/j.pmcj.2016.03.003
[20] Urcuqui-López, Christian, and Andrés Navarro Cadavid. "Framework for malware analysis in Android." *Sistemas y Telemática* 14, no. 37 (2016): 45-56. https://doi.org/10.18046/syt.v14i37.2241
[21] Utku, Anil, Ibrahim Alper Dogru, and M. Ali Akcayol. "Permission based android malware detection with multilayer perceptron." In *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pp. 1-4. IEEE, 2018. https://doi.org/10.1109/SIU.2018.8404302
[22] Wang, Wei, Zhenzhen Gao, Meichen Zhao, Yidong Li, Jiqiang Liu, and Xiangliang Zhang. "DroidEnsemble: Detecting Android malicious applications with ensemble of string and structural static features." *IEEE Access* 6 (2018): 31798-31807. https://doi.org/10.1109/ACCESS.2018.2835654
[23] Wang, Wei, Yuanyuan Li, Xing Wang, Jiqiang Liu, and Xiangliang Zhang. "Detecting Android malicious apps and categorizing benign apps with ensemble of classifiers." *Future generation computer systems* 78 (2018): 987-994. https://doi.org/10.1016/j.future.2017.01.019
[24] Yan, Ping, and Zheng Yan. "A survey on dynamic mobile malware detection." *Software Quality Journal* 26, no. 3 (2018): 891-919. https://doi.org/10.1007/s11219-017-9368-4
[25] Yang, Manzhi, and QiaoYan Wen. "Detecting android malware with intensive feature engineering." In *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 157-161. IEEE, 2016. https://doi.org/10.1109/ICSESS.2016.7883038
[26] Mas' ud, Mohd Zaki, Shahrin Sahib, Mohd Faizal Abdollah, Siti Rahayu Selamat, and Choo Yun Huoy. "A Comparative Study on Feature Selection Method for N-gram Mobile Malware Detection." *Int. J. Netw. Secur.* 19, no. 5 (2017): 727-733.
[27] Seo, Seung-Hyun, Aditi Gupta, Asmaa Mohamed Sallam, Elisa Bertino, and Kangbin Yim. "Detecting mobile malware threats to homeland security through static analysis." *Journal of Network and Computer Applications* 38 (2014): 43-53. https://doi.org/10.1016/j.jnca.2013.05.008

[28] Wei, Te-En, Hsiao-Rong Tyan, Albert B. Jeng, Hahn-Ming Lee, Hong-Yuan Mark Liao, and Jiunn-Chin Wang. "DroidExec: root exploit malware recognition against wide variability via folding redundant function-relation graph." In *2015 17th International Conference on Advanced Communication Technology (ICACT)*, pp. 161-169. IEEE, 2015. https://doi.org/10.1109/ICACT.2015.7224777

[29] Tait, Kathryn-Ann, Jan Sher Khan, Fehaid Alqahtani, Awais Aziz Shah, Fadia Ali Khan, Mujeeb Ur Rehman, Wadii Boulila, and Jawad Ahmad. "Intrusion detection using machine learning techniques: an experimental comparison." In *2021 International Congress of Advanced Technology and Engineering (ICOTEN)*, pp. 1-10. IEEE, 2021. https://doi.org/10.1109/ICOTEN52080.2021.9493543