European Journal of Information Technologies and Computer Science
Vol 4 | Issue 2 | July 2024
ISSN 2736-5492

RESEARCH ARTICLE

RetrieveDroid: k-NN Performance Evaluation of Distance Measurement Schemes for Android Malware Detection Using Case-Based Retrieval

Naomi Dassi Tchomté[1], Franklin Tchakounté[2,*], Clovis Kenmogne Tchuinte[2], and Kalum Priyanath Udagepola[3]

ABSTRACT

One of the big challenges in cybersecurity is the detection of Android attacks since Android is the most popular mobile operating system. Within this system, applications require certain permissions to access critical resources. Investigation of the use of permissions is a concern to check whether an application is not mislead to divulge sensitive information. This work aims to determine whose distance schemes offers the best malware detection based on permission similarities. Case based reasoning (CBR) is a concept which aims to find a solution based on historical experiences. CBR performance relies on finding similarities between actual cases and stored cases and then to deduce solutions. This paper proposes to transform app data as vector of appearance of dangerous permissions and to store such vectors based on CBR structure. Then we investigate k-NN classification performance related to five distance-based metrics such as Euclidean, Cosine, Manhattan, Minkowski, and Mahalanobis. Experiments were carried out with a set of 419 applications, including 203 malicious and 216 benign samples. The whole dataset has been split in training set of 291 samples with 162 benign and 129 malicious, and the testing set of 128 samples with 54 benign and 74 malicious samples. k-Nearest Neighbor (k-NN) are used as the similarity algorithm in which the distance model is varied in each of the five distance models. Results reveal that Minkowski and Manhattan models provide the best overall performance to detect Android malware based on permissions, in terms of accuracy (99.21%) and precision (97%). This work is a good start to recommend to researchers distance metrics exploitable when performing permission similarity-based detection.

Keywords: Android, Case Based Reasoning, k-NN, Permissions.

Submitted: May 03, 2024

Published: July 07, 2024

10.24018/ejcompute.2024.4.2.126

[1] Department of Computer Engineering, Institute of Technology, University of Ngaoundéré, Cameroon.
[2] Department of Mathematics and Computer Science, Faculty of Science, University of Ngaoundéré, Cameroon.
[3] Department of Information and Computer Science, Scientific Research Development Institute of Technology, Australia.

** Corresponding Authors:*
e-mail: tchafros@gmail.com, kenmogneclovis97@gmail.com

1. Introduction

Android remains the very popular mobile operating system, and forecasting reveals that it will keep this position during 2020 [1]. This popularity makes developers to propose applications to deserve services to users. Malicious people are therefore enticed to in this area. McAfee recently reported that Android malware keeps increasing in complexity and scope [2]. G Data confirms this result by revealing that during the first half of 2019, there have been more than 10000 new malicious applications per day [3]. Approaches dealing with malware detection based on permissions include static analysis [4], [5], dynamic analysis [6], and hybrid analysis [7]. Another trend consists to detect an application nature by identifying similarities with benign and malware by relying on permissions to build signatures [8]. This work investigates on distance metrics which offer the better prediction performance in case of similarity detection based on permissions. For that, the proposal relies on case based reasoning (CBR) to structure the information with permission-related variables. CBR is helpful since it allows predicting the class of an application case based on the previously stored application cases. The prediction is achieved after calculating distance-based similarity with the k-nearest neighbors (k-NN). Experiments with a set of 419 applications, including 203 malicious and 216 benign samples, reveal that Minkowski and Manhattan provide the best overall performance to

detect Android malware based on permissions in terms of accuracy (99.21%) and precision (97%).

2. Background

2.1. Permissions

On Android, resources are protected by permissions [9]. A required permission must be granted by the user if an application needs to access the related resource. This access control scheme ensures that information must stay confidential. Google classifies three types of permissions according to the risk they can make on resources [10]:

- *Normal permissions:* An application which uses normal permissions is less risky data. An example is permission to set the time zone (SET_TIME_ZONE)
- *Dangerous permissions:* An application which is granted dangerous permissions has the possibility compromise private information. This type of permission is explicitly approved by the user, otherwise the functionality is not provided. For example, READ_CONTACTS, which is the permission to read the user's contacts.
- *Signature permissions:* They include permissions granted to applications signed by the same certificate as the application that defines those permissions.

Permissions are declared in the manifest file by the developer based on application requirements. For example, if the application needs to send data to a distant server, the developer will add INTERNET permission. If the application needs to read contacts, the manifest file will include READ_CONTACTS. This work considers any type of permissions.

2.2. Case Based Reasoning

CBR is a problem-solving and learning methodology that uses similar cases that are already stored in the database, which is called case base [11]. It is based on knowledge of previous similar cases while requiring little resolution effort and associate optimization algorithms for better optimization. CBR is articulated around four major phases: retrieve, reuse, revise, and retain:

- *Retrieve:* The retrieve phase is used to find in the case base, the most similar cases of the new case. In the process of recovering similar cases, methods like k-Nearest Neighbors (k-NN) are popular. This phase is crucial in CBR because its output is exploited in the other phases. One must investigate the method applied in the retrieve phase to obtain similar cases to the new case.
- *Reuse:* This phase is also called adaptation. When the retrieve phase determines the most similar cases, the reuse phase models a scheme based on the similar cases to find the solution to the query case. For that, one can refer to majority rule, probabilistic scheme, and class-based scheme.
- *Retrieve:* The role of this phase is to re-evaluate the reuse process stage when the expected solution of the query case is not adequate. This means that if the solution found is far from reality based on expert opinions, this phase will improve.
- *Retain:* Retain is the last phase of CBR. This phase keeps in the case base the final solution along with the problem specifications/structure for exploitation for future solution prediction.

2.3. Distance Metrics

Distance metrics help learning algorithms to identify similarities between applications based its contents [12]. CBR also represents a structure that makes use of it in the retrieve phase. Formally, distance metrics are defined as distance functions providing a relationship metric between each element from a source dataset and a target dataset. There are various types of distance metrics. Only distance metrics that are within the scope of this work are presented.

2.3.1. Euclidean Distance

Euclidean distance (ED) is the most frequent distance scheme [12]. It is recommended in case data is dense and continuous, as defined in (1). M is the number of elements of the vectors x_i and x_j. Both vectors, x and y, are similar when ED is zero. They are different in case ED is higher than 0, tending to infinite.

$$ED\,(x,\ y) = \sqrt{\sum_{i=1}^{M} (x_i - y_i)^2} \tag{1}$$

2.3.2. Cosine Distance

Cosine (Cos) computes the distance between two vectors based on their separation angles (2). Two vectors are similar when their cosine value is 1. There is some similarity found when their cosine is zero and there is no similarity in case their cosine is −1.

$$Cos\,(x,\ y) = \frac{\sum_{i=1}^{M} x_i y_i}{\|x_i y_i\|} \tag{2}$$

2.3.3. Manhattan

The Manhattan distance (Man) is the sum of horizontal and vertical distance between two x and y. They are similar if Man is zero and different otherwise.

$$Man\,(x,\ y) = \sum_{i=1}^{M} |x_i - y_i| \tag{3}$$

2.3.4. Minkowski

The Minkowski distance (Min) is a generalization of the Euclidean and Manhattan distances. The Manhattan distance between x and y is the Minkowski distance with p equal to 1. The Euclidean distance is the Minkowski distance with p equal to 2.

$$Min\,(x,\ y) = \sqrt[p]{\sum_{i=1}^{M} (|x_i - y_i|)^p} \tag{4}$$

where p is a real between 1 and 2.

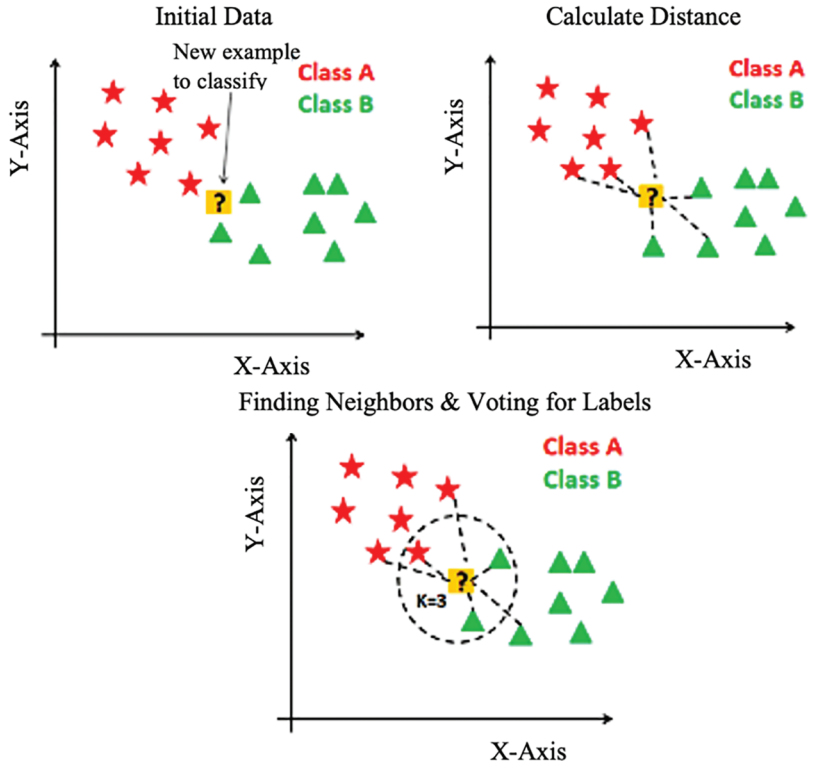

Fig. 1. kNN [17].

2.3.5. *Mahalanobis*

The Mahalanobis distance (*Mah*) calculates the gap between two data points in a multivariate space. Equation (5) corresponds to its model that measures the similarity between two different data objects. The smaller the Mahalanobis distance between vectors x and y is, the higher the similarity between x and y is [13].

$$Mah\,(x_i,\ y_i) = \sum_{i=1}^{M} (x_i - y_i)^T\, S^{-1}\, (x_i - y_i) \qquad (5)$$

where S^{-1} is the normalized covariance.

2.4. *k-Nearest Neighbor*

The k-Nearest Neighbor (k-NN) algorithm [14] is supervised learning algorithm which determines the class of an instance based on the class of the k-Nearest Neighbors. It is the most popular similarity-based retrieval technique in CBR [15], [16]. The k nearest neighbors is based on distance metrics. Algorithm 1 describes in a simple way the different steps of k-NN:

1. Loading data.
2. Initialization of the value of k.
3. For each value in the training data set.
4. VectorDistance <- calculating the metric distance (Euclidean, Manhattan, etc.) between a value of the testing data and each training data line.
5. VectorSorted <- sort vectorDistance in ascending order based on distance values.
6. Best k <- put best k rows from VectorSorted.
7. Frequent Class <- put or get the most frequent Class of these rows.
8. Return frequent Class.

Fig. 1 depicts examples using Euclidean distance. The first example shows a case with three nearest neighbors.

3. Methodology

This work aims to investigate which distance metric provides the best results to permission-based similarity in Android malware classification. It includes four phases: collection of samples, feature engineering, classification based on similarity, and evaluation of performance.

3.1. *Collection of Samples*

The first step consists to acquire datasets. We have gathered Experiments were carried out with a set of 419 applications, including 203 malicious and 216 benign samples. We have ensured that malicious and benign samples are disjointed, as well as both training and testing for malicious and normal. We proceeded using a script to check the contents and hashed of each application. We collected the malicious samples from the Android malware dataset CICAndMal2017[1] and the benign samples from Google Play.

3.2. *Feature Engineering*

The second phase consists of structuring the data, including Android applications. In this phase, an .apk is decompiled to extract the Manifest. Each application is transformed into a binary vector of 36 cells, where the cell is either 1 when dangerous permission is found within the manifest or 0 otherwise. We consider the list of 35 dangerous permissions in Table I. Their descriptions are available in the Android developer's official website [18].

[1] https://www.unb.ca/cic/datasets/andmal2017.html

TABLE I: Dangerous Permissions

1. WRITE_CALENDAR	2. READ_CALL_LOG	3. PROCESS_OUTGOING_CALLS
4. GET_ACCOUNTS	5. ACCESS_FINE_LOCATION	6. READ_PHONE_STATE
7. READ_PHONE_NUMBERS	8. CALL_PHONE	9. ANSWER_PHONE_CALLS
10. SEND_SMS	11. RECEIVE_SMS	12. READ_SMS
13. RECEIVE_WAP_PUSH	14. RECEIVE_MMS	15. READ_EXTERNAL_STORAGE
16. WRITE_EXTERNAL_STORAGE	17. CHANGE_CONFIGURATION	18. DELETE_PACKAGES
19. DISABLE_KEYGUARD	20. GET_PACKAGE_SIZE	21. GET_TASKS
22. RECORD_AUDIO	23. RESTART_PACKAGES	24. WRITE_APN_SETTINGS
25. WAKE_LOCK	26. VIBRATE	27. SET_WALLPAPER
28. INTERNET	29. INSTALL_PACKAGES	30. CHANGE_WIFI_STATE
31. CHANGE_NETWORK_STATE	32. READ_LOGS	33. WRITE_CONTACTS
34. WRITE_SETTINGS	35. WRITE_CALL_LOG	

TABLE II: Case Base Representation

	$Perm_1$	$Perm_2$	...	...	...	...	$Perm_{35}$	Malicious/benign
Case 1	0	1	...	...	...	...	...	...
...	...	...	0	...	...	...	...	...
...	...	...	...	...	1	1	1	...
Case n	0	1	...	...	...	...	...	...

The case base is represented in Table II. A case is an application or sample, and an attribute is one of the 35 permissions represented by their presence or absence in the application. The solution is the decision of whether the application is benign or malicious.

3.3. *Classification Based on Distance Similarity*

The feature engineering results are inserted into a .csv file. A case represents a vector of an application. The solution of a case refers to the class of that application (malicious or benign). The k-NN Python script is parameterized with the different distance metrics to compute similarity measures in the CBR retrieve. For prediction, the whole dataset has been split in a training set of 291 samples with 162 benign and 129 malicious, and the testing set of 128 samples with 54 benign and 74 malicious samples. The next section presents and discusses the results. During this phase, the training cases constitute the case base where testing samples are considered as query cases, i.e., cases to look for solutions.

4. Evaluation of Performance

This section is dedicated to analyze classification results based on the different distance metrics. For that, two metrics are computed for each distance metric: precision and accuracy. The precision is the proportion of positive/negative identifications, which was actually correct[2]. There are precision can be evaluated for positive (6) and negative (7) predictions:

$$Precision_{positive} = \frac{TP}{TP + FP} \tag{6}$$

$$Precision_{negative} = \frac{TN}{TN + FN} \tag{7}$$

The accuracy is the fraction of correct predictions[3].

$$Accuracy = \frac{TP}{TP + FP} \tag{8}$$

Table III presents the results obtained after applying 3-NN with the brute-force search, which consists of systematically enumerating all possible candidates for the solution and checking whether each candidate verifies the problem statement.

We observe in Fig. 2 that the Manhattan and Minkowski models give the same results and present the best performances when precision and accuracy are simultaneously considered. These distance models are able to correctly predict negative and positive with an average precision of 97% and an accuracy of 99.21%. Euclidean distance is perfectly precise in recognizing benign applications, whereas Cosine distance is perfectly precise in recognizing malicious applications. Cosine and Mahalanobis are less accurate in discriminating between malicious and benign applications due to the fact that they require more information to compute similarity. They are both lazy because they fail to be precise in recognizing 46.40% and 36.10%, respectively, of benign applications. These rates are higher according to the size of the datasets.

5. Related Works

Android security based on permissions usage falls within several axes. The first axe concerns analysis of permissions declared in the manifest and those required by the application. Authors try to determine whether permissions are over privileged, to determine suspicious combinations of permissions [5], [19], and to look for vulnerable permission API [20] and permission graph construction [21]. The second axe concerns scrutinizing permission API accessed during runtime, refining Android access control systems

[2] https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall

[3] https://developers.google.com/machine-learning/crash-course/classification/accuracy

TABLE III: PERFORMANCE METRICS

	3-NN		
	$Precision_{positive}$	$Precision_{negative}$	Accuracy
Manhattan	98.60%	96.30%	99.21%
Minkowski	98.60%	96.20%	99.21%
Euclidean	98.60%	100%	97.65%
Cosine	100%	56.60%	85.90%
Mahalanobis	0.98	63.90%	84.37%

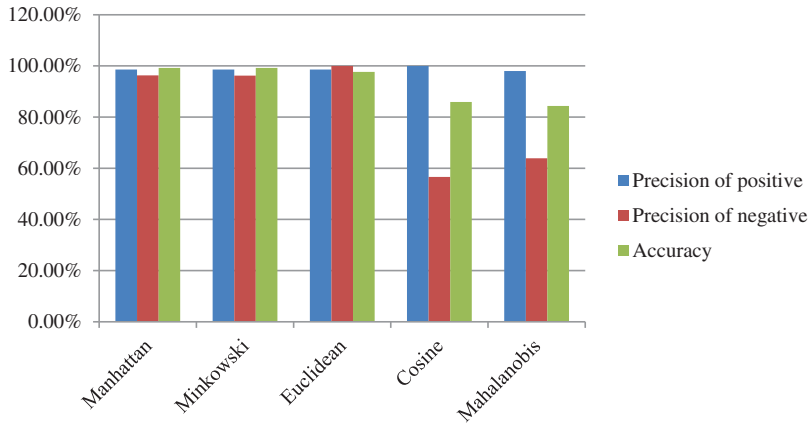

Fig. 2. Results.

[10], and coupling machine learning algorithms to enrich knowledge [22]–[24]. The third axe refers to put in place mechanisms to assist developers to set permissions more effectively and accurately [25], [26]. The fourth axe aims to determine and raise risks incurred when an application has certain permissions [27]. The fifth axe includes permissions to build signatures exploited to find similarities among applications [8]. Complementary to the previous works, this research aims to investigate which distance metrics are suitable to improve permission-based detection results based on similarities.

6. Conclusion and Perspectives

A trend in cyber-security consists of determining the nature of an application by computing similarities against signatures of benign and malicious. These signatures include static and dynamic traits. This research investigated distance metrics which provide better similarity performance with permission-based signatures. To achieve it, the data has been structured under a case based reasoning scheme in which cases are made of vectors of permissions presence. The retrieve has been executed with five distance metrics: Euclidean, Cosine, Manhattan and Minkowski, and Mahalanobis. The similarity measurement has been captured using k-NN with various distance models. A collection of 419 application samples has been used in experiments. The whole dataset has been split into 203 malicious and 216 benign applications. Results revealed that Minkowski and Manhattan models provide the best overall performance to detect Android malware based on permissions in terms of accuracy (99.21%) and precision (97%). Additionally, it has been observed that Cosine is specifically adequate for identifying malicious applications in similarity measurement based on permissions, whereas Mahalanobis is specifically adequate for identifying benign applications in similarity measurement based on permissions. This work has been found relevant to guide researchers on the detection of malware relying on similarities. However, this work requires a step forward. As a perspective, we will make the same investigation by coupling permissions with other static features such as API and code graph.

7. Recommendations

Due to the fact that Manhattan and Minkowski are much more precise and accurate, they are recommended to researchers willing to efficiently identify both benign applications and malicious applications relying on similarity schemes on permissions. If one needs to recognize only malicious applications with similarity based, cosine applied to permissions is recommended. If one needs to recognize only benign applications with similarity based, Euclidean applied to permissions is recommended.

8. Limitations

This work is principally limited because it focused exclusively on permissions while there are other elements, such as API_CALL and intents, that can be exploited to improve similarity evaluation.

Conflict of Interest

The authors declare that they do not have any conflict of interest.

References

[1] Statista. *Tablet operating systems' market share worldwide from 2013 to 2020*. 2018. [Online]. Available from: https://www.statista.com/statistics/272446/global-market-share-held-by-tablet-operating-systems/. [Accessed: 11-Jan-2020].

[2] McAfee. McAfee mobile threat report. 2019.

[3] GDATA. *Mobile Malware Report-no let-up with Android malware*. 2019. [Online]. Available from: https://www.gdatasoftware.com/news/2019/07/35228-mobile-malware-report-no-let-up-with-android-malware. [Accessed: 11-Jan-2020].

[4] Kabakus AT. What static analysis can utmost offer for android malware detection. *Inf Technol Control*. 2019;48(2):235–49. doi: 10.5755/j01.itc.48.2.21457.

[5] Li J, Sun L, Yan Q, Li Z, Srisa-An W, Ye H. Significant permission identification for machine-learning-based android malware detection. *IEEE Trans Ind Inform*. 2018 Jul;14(7):3216–25. doi: 10.1109/TII.2017.2789219.

[6] Thanigaivelan NK, Nigussie E, Hakkala A, Virtanen S, Isoaho J. CoDRA: context-based dynamically reconfigurable access control system for android. *J Netw Comput Appl*. 2018 Jan;101:1–17. doi: 10.1016/j.jnca.2017.10.015.

[7] Deepti V, Choudhary SP, Subrata R, Kumar CRS. Malware detection by static checking and dynamic analysis of executables. *Int J Inf Secur Priv*. 2017;11(3):29–41.

[8] Xue Y, an Tan Y, Liang C, Li Y, Zheng J, Zhang Q. RootAgency: a digital signature-based root privilege management agency for cloud terminal devices. *Inf Sci (Ny)*. 2018 May;444:36–50. doi: 10.1016/j.ins.2018.02.069.

[9] Shrivastava G, Kumar P, Gupta D, Rodrigues JJPC. Privacy issues of android application permissions: a literature review. *Trans Emerg Telecommun Technol*. 2019 Oct;31(12):1–17. doi: 10.1002/ett.3773.

[10] Niu S, Huang R, Chen W, Xue Y. An improved permission management scheme of android application based on machine learning. *Secur Commun Netw*. 2018 Oct;2018:1–12. doi: 10.1155/2018/2329891.

[11] Tchomté N, Asghar S, Javaid N, Dayang P, Danga D, Oyono D. A case based reasoning coupling multi-criteria decision making with learning and optimization intelligences: application to energy consumption. *EAI Endorsed Trans Smart Cities*. 2018 Jul;162292. doi: 10.4108/eai.26-6-2018.162292.

[12] Abu Alfeilat HA, Hassanat ABA, Lasassmeh O, Tarawneh AS, Alhasanat MB, Eyal Salman HS, *et al*. Effects of Distance Measure Choice on K-Nearest Neighbor Classifier Performance: A Review. *Big Data*. 2019;Dec;7(4):221–48. doi: 10.1089/big.2018.0175. Epub 2019 Aug 14. PMID: 31411491.

[13] Zhang H, Zhou Y, Feng D. Mahalanobis distance similarity measure based distinguisher for template attack. *Secur Commun Netw*. 2015 Mar;8(5):769–77. doi: 10.1002/sec.1033.

[14] Zhang Z. Introduction to machine learning: k-nearest neighbors. *Ann Transl Med*. Jun 2016;4(11):1–7. doi: 10.21037/atm.2016.03.37.

[15] Portinale L. Exploiting Markov random fields to enhance retrieval in case-based reasoning. *The Thirty-Second International Flairs Conference*, Sarasota, Florida, USA, 2019.

[16] Choudhury N, Begum SA. *Neuro-Fuzzy-Rough Classification for Improving Efficiency and Performance in Case-Based Reasoning Retrieval*. Singapore: Springer; 2020, pp. 29–38.

[17] Navlani A. *KNN classification using Scikit-learn. Learn K-Nearest Neighbor (KNN) classification and build KNN classifier using Python Scikit-learn package*. 2018. [Online]. Available from: https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn. [Accessed: 11-Jan-2020].

[18] AndroidDevelopers. *Manifest.permission*. 2019. Available from: https://developer.android.com/reference/android/Manifest.permission. [Accessed: 11-Jan-2020].

[19] Arora A, Peddoju SK, Conti M. PermPair: android malware detection using permission pairs. *IEEE Trans Inf Forensics Secur*. 2019;51:1968–82. doi: 10.1109/TIFS.2019.2950134.

[20] Liu X, Liu J, Zhu S, Wang W, Zhang X. Privacy risk analysis and mitigation of analytics libraries in the android ecosystem. *IEEE Trans Mob Comput*. 2019 Mar:91(5):1184–99. doi: 10.1109/tmc.2019.2903186.

[21] Liu Y, Wang SC, Yang Y, Chen YC, Sun HM. An automatic UI interaction script generator for android applications using activity call graph analysis. *Eurasia J Math Sci Technol Educ*. 2018;14(7):3159–79. doi: 10.29333/ejmste/91668.

[22] Arslan RS, Doğru IA, Barişçi N. Permission-based malware detection system for android using machine learning techniques. *Int J Softw Eng Knowl Eng*. 2019 Jan;29(1):43–61. doi: 10.1142/S0218194019500037.

[23] Yildiz O, Doğru IA. Permission-based android malware detection system using feature selection with genetic algorithm. *Int J Softw Eng Knowl Eng*. 2019 Feb;29(2):245–62. doi: 10.1142/S0218194019500116.

[24] Tchakounté F, Hayata F. Supervised learning based detection of malware on android. In *Mobile Security and Privacy: Advances, Challenges and Future Research Directions*. Elsevier Inc, 2017, pp. 101–54.

[25] Bahrini M, Wenig N, Meissner M, Sohr K, Malaka R. HappyPermi. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–6, 2019. doi: 10.1145/3290607.3312914.

[26] Xu G, Xu S, Gao C, Wang B, Xu G. PerHelper: helping developers make better decisions on permission uses in android apps. *Appl Sci*. 2019 Sep;9(18):3699. doi: 10.3390/app9183699.

[27] Alshehri A, Marcinek P, Alzahrani A, Alshahrani H, Fu H. Puredroid: permission usage and risk estimation for android applications. In *ACM International Conference Proceeding Series*, Houston TX USA, 2019, pp. 179–84. doi: 10.1145/3325917.3325941.