CAFM: Precise Classification for Android Family Malware

Dan Li, Runbang Pan, Ning Lu, and Wenbo Shi(✉)

School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China
shiwb@neuq.edu.cn

Abstract. Family malware classification is becoming progressively urgent because of the increasing diversity of family malware and the different hazards it causes. There is a growing concern that classification is at a disadvantage owing to its problems. For one thing, obtaining the crucial features of innumerable families is arduous. For another, constructing a classification model that fully learns multi-class samples is intricate. To solve these problems, it proposes a precise classification for Android family malware called CAFM in this paper. It profoundly analyzes the relationship between the information implicit in features and the degree of differentiation among families. We select the features containing context information as feature representations. In addition, it employs a specially designed deep neural network model with upgraded learning capability for grasping the continuous features of family malware utterly. Experimental verification on a real-world dataset shows that the CAFM can effectively implement family classification, and the classification accuracy reaches 97.73% when the length of the opcode sequence is 700. Compared with other classifiers, the Kappa coefficient of the comprehensive evaluation indicator also reached 0.9725 and is at least 0.1225 higher than comparison classifiers.

Keywords: Android malware · Family · Classification

1 Introduction

Recent developments in the diversity of family malware have heightened the desideratum for Android mobile phone's security [1]. Users are at increased risk, which puts greater demands on anti-virus organizations and security researchers. For example, Kaspersky's mobile threat types announced in 2021, Adware, and

This work was supported by the National Natural Science Foundation of China (Nos. 62072092, 62072093 and U1708262); the China Postdoctoral Science Foundation(No. 2019M653568); the Key Research and Development Project of Hebei Province (No. 20310702D); the Natural Science Foundation of Hebei Province (No. F2020501013); the Fundamental Research Funds for the Central Universities (No. N2023020).

W. Shi et al. (Eds.): SPNCE 2021, LNICST 423, pp. 382–394, 2022.
https://doi.org/10.1007/978-3-030-96791-8_28

Risktool accounted for 57.26% and 21.34% respectively, ranking first and second [2]. Worm, which is at the end of the list, accounts for 0.01%. From a quantitative standpoint, it indicates that the Adware and Risktool types require more attention than Worm from anti-virus organizations and security researchers. Anti-virus organizations must not only implement benign and malicious classifications for malware but also further classify family malware. Research on family-level malware can concentrate more on widely spread and highly threatened families, rather than individual samples or families with less risk. Therefore, it is necessary to propose a precise classification method for family malware.

Family malware classification is to classify a group of malware with common peculiarities and behaviors together, and the members of the group share some unique peculiarities within the family [3]. It is a multi-classification problem. The central purpose is to construct a mapping model that reflects the idiosyncrasies and classes within the family. The model can predict the new sample and procure its family label. Given innumerable families and masses of software candidate features, family classification faces two technical challenges.

Challenge 1: Difficulty in obtaining features that contain enough family representative peculiarity information. Family malware is innumerable and varies in its features. It is necessary to select the feature subset that can represent innumerable families' behavior characteristics. Existing features analysis can be divided into runtime features and unpacking features in the light of different acquisition methods. Runtime features require actual or simulated running programs to obtain feature information, which consumes numerous time and computing resources [4–6]. It is inapposite for anti-virus organizations. Among the unpacking features, a single feature such as permission cannot truly represent the behavior of malware because of excessive permission requests [7]. The n-gram opcode feature will cause the feature subset to increase exponentially as the value of n increases, causing the explosion of the feature space [8]. Therefore, choosing the appropriate features is the first technical problem.

Challenge 2: Difficulty in constructing a classification model that adequately learns the peculiarities of each family. The diversity of families leads to different calling relationships to implement malicious behavior. The crucial malicious instructions or code of each family are not the same on the same feature. The signature method needs to match the signature library and cannot recognize malware that is not in the signature library [9]. Shallow machine learning methods are more suitable for processing a limited number of samples rather than large-scale datasets [10,11]. It can only learn shallow exhibitions but can not acquire high-level abstract features. Therefore, building a classification model that utterly memorizes the features of multi-class samples is the second technical challenge.

Aiming at these challenges, a precise family malware classification method is proposed in this paper. The behavioral idiosyncrasies of each family and the correlation between the features are analyzed, and the features containing as much information as possible are utilized to acquire more accurate depictions. In addition, to construct a classification model with sufficient learning samples and

consider the requirements of anti-virus organizations, a one-dimensional deep neural network (1d-CNN) is adopted with automatically procuring the internal patterns of features. It can not only reduce the information loss of feature data conversion but also retain the pivotal information to the maximum extent.

The main contributions presented in this work are summarized as follows:

(1) To obtain representative features, the feature expressions in different families are analyzed, and the feature heterogeneity in diverse families is mined.
(2) A precise deep network classification model for automatic learning of internal patterns is constructed. The effective automatic extraction of the input features is realized.
(3) Experimental results show that the proposed CAFM performs well on a real-world dataset, with an accuracy of 95.45% or above when the length of opcode sequence are 700 and 5000, which is at least 5.56% higher than the comparative classifiers.

The remainder of this paper is structured as follows. Section 2 is an introduction to the feature extraction process and an explanation of the problem statement. Section 3 describes the overall framework of the proposed CAFM, the composition of each component, and the corresponding functions. Section 4 is the analysis of experimental results, which verifies the classification performance of the CAFM. Section 5 is to sort out and summarize the related literature. The conclusion is in Sect. 6.

2 Feature Extraction and Problem Statement

This section introduces the process of feature extraction, defines the family malware classification problem, and gives the formal expression of the problem.

2.1 Feature Extraction

The feature extraction is to obtain candidate features that can represent family behavior from the representable forms of a striking number of applications. It is extracted in a specific form for further processing.

The Android package is a specific packaging form of the Android application. The malicious behavior is realized by the function developed by the attacker, which needs to call the application programming interface (API). Therefore, the unpacking feature analysis can be realized by decomposing the source package to obtain specific features. The implementation of malware needs to call specific APIs to complete. For example, malicious billing software will call the API for sending SMS, and privacy stealing software will call the API for accessing the address book. Dalvik is a virtual machine designed by Google for the Android platform, and the dalvik instruction set contains operational information about the application and can be detected by analyzing dalvik opcode information.

Classes in the program code will generate smali files in the appropriate directories. Each smali file contains a format statement, and the statements in smali follow a set of grammar specifications. In the decompiled smali file is a dalvik command. In reality, the smali file is an explanation for the dalvik virtual machine. It has more than 200 dalvik instructions, each of which points to the operation of the register. The opcode feature is derived from the dalvik instruction in the smali file. The preprocessing process of acquiring APIs and opcode sequences consists of three important steps:

(1) Decompile the .apk file to get the smali file.
(2) Extract the dalvik opcodes and API from the smali file.
(3) Obtain opcode sequence according to the Android opcode constant list and API list.

2.2 Problem Statement

The opcode sequences and APIs obtained after feature extraction are the features to be analyzed. The family malware classification is represented by learning the features of the opcode sequences and APIs to obtain the predicted results. It will explain by problem definition and problem decomposition.

Problem Definition. The family malware classification is to classify the unknown attribute samples into the class of the family, that is, to give the class label of the family of the detected samples. Through the training set $\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$, a mapping f from the input space X to the output space Y is established, where $Y > 2$.

$$f : X \to Y \tag{1}$$

Problem Decomposition. The classification problem is to firstly build a classification model and then utilize it to make predictions. Therefore, the family malware classification can decompose into two subproblems.

Subproblem 1: Feature depiction. After feature extraction, it requires to transform into a data form that the classification model can recognize. It should not only contain as much information as possible but also facilitate model processing.

Subproblem 2: Feature memorizing and sample prediction. The classification model is constructed to obtain the internal abstract representation of features. The opcode sequences and APIs containing context and sequence information are directly input into the model for feature learning without other redundant data transformations. With the continuous iteration and parameter update of the model, new samples can be predicted after the feature learning is completed.

In summary, subproblem 1 is to process the acquired original features to get a more appropriate representation to express more information. Subproblem 2 is the construction of the classification model, which can thoroughly learn the abstract representation of multiple families and obtain the high-level expression of the features.

3 CAFM

This section describes the overall framework of CAFM, the assembly modules, and the corresponding functions of each module. It includes the method of obtaining the representation of representative features and the construction of the family classification model.

3.1 Overall Framework

Given the problems faced by the family classification, the CAFM proposes. Figure 1 illustrates its overall framework, which consists of two major modules. One is feature depiction, and the other is feature memorizing and sample prediction.

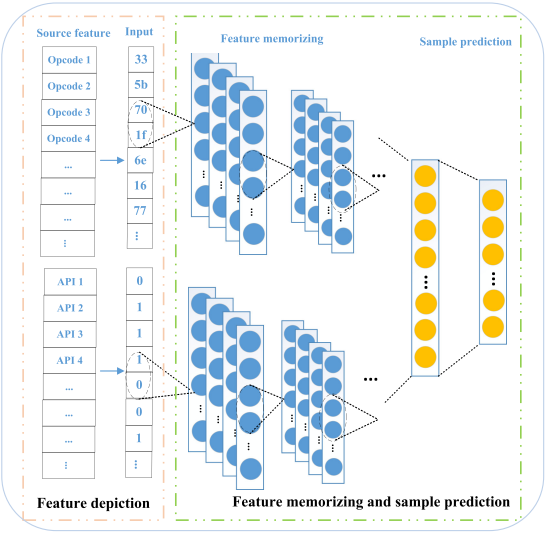


Fig. 1. The overall framework of the CAFM.

Feature Depiction. This module is to optimally represent the extracted opcode sequences and APIs features. It allows the opcode and API information to be included as much as possible while forming a corresponding data format for the input of the classification model.

Feature Memorizing and Sample Prediction. This module is utilized for training features and predicting unknown samples. Training is the stage of learning and memorizing features. The model obtains the high-level feature representation of features. The completion of training represents the formation of a classification model, which can predict the label of unknown samples.

3.2 Feature Depiction

Feature depiction is to select the pivotal elements of the extracted features and represent them in a format that can be recognized by the classification model. On one hand, it is to reduce the computation and resource consumption caused by the dimensionality of features. On the other hand, it is to reduce redundant information and obtain crucial features.

The extracted opcode sequences and APIs can not be directly sent to the model for training; consequencely, the crucial sub-features need to be represented, and the appropriate data format needs to be selected. The opcode sequences include consecutive opcode information and relative position information. API features are selected based on their frequency of occurrence in the data set, from high to low. The steps for feature depiction are as follows:

Acquire original features. The original opcode sequence is opcode 1, opcode 2, opcode 3,$\cdots$. The APIs are API 1, API 2, API 3,$\cdots$.

Coding. It will encode each opcode by that total number to get its fixed sequence number in the opcode dictionary. APIs have a fixed number in the API dictionary in descending order of frequency.

Representation. The extracted opcode sequence and the location corresponding to the API are used for data form representation. The range of each opcode is 0–256, which is a decimal number. The value of the API is 0 or 1, and the position of the API is set to 1 when this call occurs for the sample; otherwise, it is 0.

3.3 Feature Memorizing and Sample Prediction

Feature memorizing is the learning process of the classification model for feature knowledge, and sample prediction is the label prediction using the learned classification model. Both are momentous steps in the realization of classification for family malware.

To learn the internal patterns of the features more accurately, this paper chooses the 1d-CNN with two channels as the network for automatically learning the high-level abstract features [12]. This choice is based on two considerations. The first is to improve the ability of the network to obtain information. Two channels can input different features to obtain feature information from different angles in the family, providing more information. The second is to obtain the original feature information to a greater extent. The opcode sequence itself contains the sequence of the called opcodes, which includes context information and relative location characteristics. This is closer to the data format of a one-dimensional signal. The 1d-CNN is essentially the same as the convolutional neural network (CNN). It also has the advantages of CNN's translation invariance for feature recognition, and the one-dimensional large convolution kernel will not bring too many parameters and calculations. The eigenvalues of the sequence can be obtained more comprehensively while suppressing overfitting. The structure of the two-channel 1d-CNN includes two identical series of convolutional layers and pooling layers, and finally, the result is output through the fully connected layer. The main steps are as follows:

Feature Memorizing Stage. *Convolutional operation:* The convolutional layer is an important part of the CNN, which can extract highly abstract features through the convolutional operation. The equation for the convolutional operation is as follows:

$$Cov(x, y) = \sum_{a=0}^{w} F(a) \times G(x - a) \tag{2}$$

Pooling operation: The feature with spatial invariance is obtained by reducing the resolution of the feature map. When it faces the sparse feature, it can reduce the mean shift of the estimation caused by the parameter error of the convolution layer. The equation is as follows:

$$P = \max_{w}\{A^l\} \tag{3}$$

Integration: Each neuron in the fully connected layer is fully connected to all neurons in the previous layer. The goal is to integrate features from convolutional and pooling operations.

$$C = P^O \bigoplus P^A \tag{4}$$

where P^O and P^A are outputs of opcode channels and API channels, respectively. The above equation indicates that the output C of the integration layer is obtained by connecting the results of the two channels. The output C is utilized as the input of the classification layer, and the loss of this training is output through the classifier.

Sample Prediction Stage. All parameters have been obtained in the previous stage. This stage only needs to input the vectorized data from the input layer to realize predictions.

4 Experiment Studies

To verify the performance of the proposed CAFM, this section verifies it by the real-world dataset. The reliability and classification performance of the proposed CAFM is verified from different perspectives.

4.1 Experiment Setup

The experiment setup is expanded from two parts, including the description of the dataset involved in the experiment and the evaluation indicators adopted for multi-classification.

Evaluation Indicators. The evaluation indicator is a multi-classification standard, mainly including confusion matrix, F1-micro, F1-macro, Kappa, accuracy, and hamming distance.

Dataset Description. The dataset adopted in this paper includes benign software and malicious software, a total of 8791 samples from the Chinese application market and AMD datasets respectively [13]. These samples are divided into seven families, a benign comprehensive family (called pos family) and six malware families. The pos family is a collection of randomly selected benign samples, which is set as a family here regardless of type. The number of varieties contained in each family is also presented in Table 1.

Table 1. Dataset information.

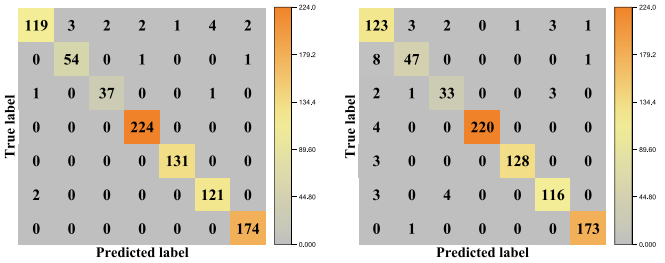
Family	Varieties	Numbers
BankBot	8	637
DroidKungFu	6	546
FakeInst	5	2156
Fusob	2	1203
Kuguo	1	1189
Mecor	1	1820
Pos	–	1240

4.2 Results Analysis

To verify the classification performance of the proposed CAFM, the experimental results are presented and analyzed in this section. It is mainly developed from two aspects, one is the elaboration of the multi-classification performance of the family malware, and the other is the comparison and analysis of the classification results of CAFM and other classifiers.

Performance of Malware Familial Classification. The family classification performance of the proposed CAFM is mainly demonstrated from two perspectives: confusion matrix and the distribution of opcode sequence lengths for different malware families. In the experiment, the ratio of the training set, test set, and verification set is 8:1:1, respectively.

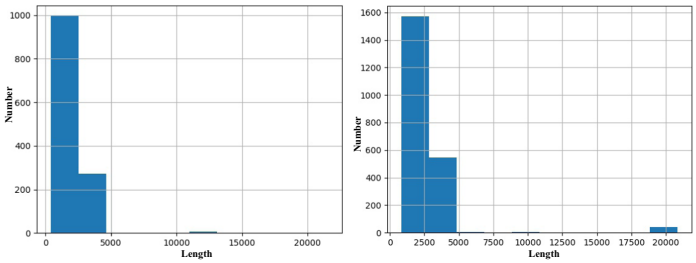
As shown in Fig. 2, the classification results of CAFM with an opcode length of 700 and 5000 are displayed. The correct classification samples in the confusion matrix are distributed diagonally from the top left to the bottom right. The smaller the value outside the diagonal, the better the classification result. It can be seen that the classification of most families is correct despite the uneven data sample. Only a few families with a large number of variants showed a small number of misclassified samples. When the length of the opcode sequence is 700, the classification effect is better, and only a few samples are misclassified in the face of the family with the highest similarity in the eight variants. The F1-micro and Kappa coefficients reach 0.9773 and 0.9725, respectively, which shows a favorable family classification effect.



(a) The results on opcode700. (b) The results on opcode5000.

Fig. 2. The confusion matrix of CAFM on different opcode sequences length.

In addition, to further explore the relationship between opcodes and samples, the length of the opcodes sequence is statistically analyzed. The distribution of opcode length of different families is obtained. The similarity between families in the length of opcode sequences is verified. As shown in Fig. 3, the distribution of opcode sequence length is given by taking Fusob and FakeInst families as examples. The abscissa axis is the length of the opcode sequence, and the ordinate is the number of samples. It can be seen that the same family presents a certain regularity in the distribution of opcodes numbers. The length of the Fusob family opcode sequence is shorter than other families, and the basic length is less than 5000. The FakeInst family, which includes five varieties, has a wide range of length variations, with some samples appearing at 20,000. These all demonstrate similarities in the length of opcode sequences within the same family or within the same variety, reflecting similar malevolent behavior that might be possible.



(a) The Fusob family. (b) The FakeInst family.

Fig. 3. The opcode sequences length distribution of malware family.

Performance Comparison with Different Classifiers. To further verify the classification effect of the proposed CAFM, this paper utilizes more abundant and multi-angle classification model evaluation indicators to illustrate the experimental data. We adopt the feature of n-gram as the contrast feature and employ different classifiers to carry out a multi-indicator comparison.

Since n-gram opcode is a widely utilized analysis method based on opcode features, this paper chooses opcode analysis methods with different n values as the main comparison feature. Here we take the 2-gram opcode as an example because research shows that the increase of n may cause an explosion in the data space. Meanwhile, the classification performance does not increase with the increase of excessive n value. Many studies show that the effect is better when the value of n is 2. The classifiers are implemented by combining 2-gram opcode features with 1d-CNN, support vector machine (SVM), random forest (RF), decision tree (DT), naive Bayes (NB), and k-nearest neighbor (KNN) [14].

Figure 4 and Fig. 5 show the results of each classifier on different classification indicators. In terms of the Kappa coefficient in Fig. 4, CAFM reaches 0.9449, which is 0.0665 higher than 0.8784 of the highest RF among other classifiers. On the F1-micro, the RF reaches 0.8989 and performs well, but it is still 0.0556 lower than CAFM. On F1-macro, CAFM is 0.0615 higher than the optimal RF. Figure 5 shows the classification effect of each classifier on the hamming distance indicator. The smaller the value, the better the classification effect. Similarly, CAFM reached the smallest value of 0.0455, and the NB with the worst classification effect reached 0.4625, proving the obvious advantage of CAFM on this indicator. In short, superior results are achieved with the proposed CAFM in various indicators. The comprehensive classification indicators F1-micro, F1-macro, and Kappa have all achieved more than 0.9317, which is an ideal multi-classification effect.

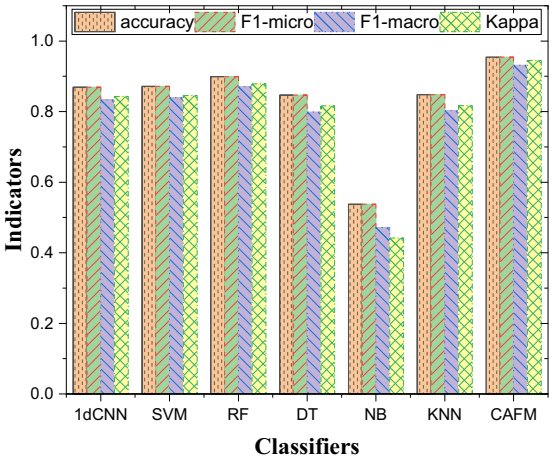


Fig. 4. The results of each classifier on different classification indicators.

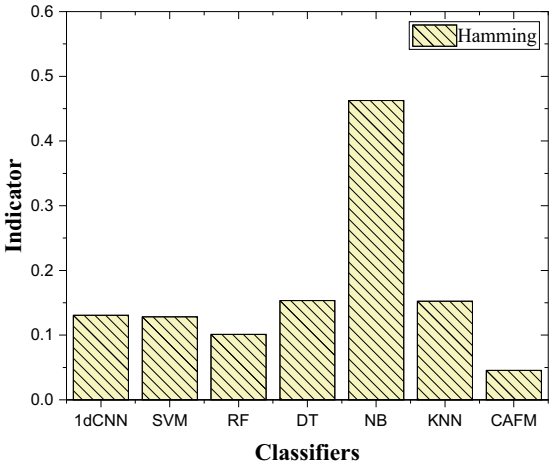


Fig. 5. The results of each classifier on hamming indicators.

5 Related Literature

Family malware classification is mainly divided into run-time feature-based and unpacking feature-based analysis methods in conformity with different feature extraction modes. This section describes the related literature from these two perspectives for illustration.

Run-Time Feature-Based Analysis. For example, Martin, A. et al. [6] proposed a classification tool that combines the form of state sequences of run-time features with Markov chains. Feng, P. et al. [5] proposed to implement malware detection based on multiple types of run-time behavior features. The feature selection algorithm was employed to remove noise or irrelevant features and extract key behavior features. Dhalaria, M. [4] and Cavli, O.F.T. et al. [15] proposed a method to combine unpacking and run-time malware analysis features. It provided an effective way to solve malware detection. D'Angelo, G. et al. [16] utilized unpacking and run-time features, including hash fingerprints, permissions, application info, network flows, and dynamic API calls, to describe the behavior peculiarities of the software. These methods are expensive in terms of computation and resource consumption. It is not befitting for the user requirements of anti-virus organizations.

Unpacking Feature-Based Analysis. For instance, the authors constructed the fingerprint of the malware families using n-grams analysis and features hashing [9]. In [17], the authors generated a fingerprint for each family. This analysis method relies on a signature library and cannot identify samples that are not in the library. Yuan, H. et al. [11] proposed an unpacking feature-based detection method based on tf-idf and machine learning, using the tf-idf algorithm to calculate the number of permissions used. Turker, S. et al. [10] proposed a framework that extracts requested permissions and API calls from Android malware

samples and used them as features to train a large number of machine learning classifiers. This analysis method cannot learn deep features in view of its limitations. Alswaina, F. et al. [7] implemented a reduced permission set and fed it into a machine learning algorithm for classification. This analysis method may be subject to abuse of permission requests. Opcode feature analysis methods such as Gaviria de la Puerta, J. et al. [8] and Kang, B. et al. [18] used n-gram features combined with machine learning. This analysis method causes the data space to explode when the value of n is too large. Accordingly, these unpacking-based feature analysis methods are not considered.

Unlike the above methods, the feature representation method proposed in this paper can effectively express the diversity and behavior characteristics of software families. The two channels 1d-CNN adopted by CAFM can more fully acquire the input features and utterly learn the high-level abstract expression of the features.

6 Conclusion

This paper proposes a precise classification method for family malware. Through in-depth analysis of features, feature subsets are obtained, which can represent family diversity. The 1d-CNN with two channels is designed, which can not only utilize the original feature data format to a greater extent but also preserve the information of the source features. The design of two channels fully learns the internal abstract pattern of fusion features and provides a more accurate model for family classification.

References

1. Mercaldo, F., Santone, A.: Formal equivalence checking for mobile malware detection and family classification. IEEE Trans. Softw. Eng. (2021)
2. Kaspersky: mobile malware evolution 2020 (2021)
3. Chakraborty, T., Pierazzi, F., Subrahmanian, V.S.: Ec2: Ensemble clustering and classification for predicting android malware families. IEEE Trans. Dependable Secure Comput. **17**, 262–277 (2020)
4. Dhalaria, M., Gandotra, E.: A hybrid approach for android malware detection and family classification. Int. J. Interact. Multimedia Artif. Intell. **6**(6) (2021)
5. Feng, P., Ma, J., Sun, C., Xu, X., Ma, Y.: A novel dynamic android malware detection system with ensemble learning. IEEE Access **6**, 30996–31011 (2018)
6. Martín, A., Rodríguez-Fernández, V., Camacho, D.: CANDYMAN: Classifying android malware families by modelling dynamic traces with Markov chains. Eng. Appl. Artif. Intell. **74**, 121–133 (2018)
7. Alswaina, F., Elleithy, K.: Android malware permission-based multi-class classification using extremely randomized trees. IEEE Access **6**, 76217–76227 (2018)
8. de la Puerta, J.G., Sanz, B.: Using Dalvik opcodes for malware detection on android. Logic J. IGPL **25**(6), 938–948 (2017)
9. Zhang, L., Thing, V.L., Cheng, Y.: A scalable and extensible framework for android malware detection and family attribution. Comput. Secur. **80**, 120–133 (2019)

10. Türker, S., Can, A.B.: Andmfc: android malware family classification framework. In: 2019 IEEE 30th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC Workshops). pp. 1–6. IEEE (2019)
11. Yuan, H., Tang, Y., Sun, W., Liu, L.: A detection method for android application security based on TF-IDF and machine learning. PLoS ONE **15**(9), e0238694 (2020)
12. Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., Inman, D.J.: 1d convolutional neural networks and applications: a survey. Mech. Syst. Signal Process. **151**, 107398 (2021)
13. Wei, F., Li, Y., Roy, S., Ou, X., Zhou, W.: Deep ground truth analysis of current android malware. In: Polychronakis, M., Meier, M. (eds.) DIMVA 2017. LNCS, vol. 10327, pp. 252–276. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-60876-1_12
14. Pedregosa, F., et al.: Scikit-learn: machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
15. Cavli, O.F.T., Sen, S.: Familial classification of android malware using hybrid analysis. In: 2020 International Conference on Information Security and Cryptology (ISCTURKEY), pp. 62–67. IEEE (2020)
16. D'Angelo, G., Palmieri, F., Robustelli, A., Castiglione, A.: Effective classification of android malware families through dynamic features and neural networks. Connection Sci. 1–16 (2021)
17. Massarelli, L., Aniello, L., Ciccotelli, C., Querzoni, L., Ucci, D., Baldoni, R.: Android malware family classification based on resource consumption over time. In: 2017 12th International Conference on Malicious and Unwanted Software (MALWARE), pp. 31–38. IEEE (2017)
18. Kang, B., Yerima, S.Y., Sezer, S., McLaughlin, K.: N-gram opcode analysis for android malware detection. arXiv preprint arXiv:1612.01445 (2016)