Journal of Engineering and Applied Sciences 14 (23): 8761-8768, 2019
ISSN: 1816-949X

A Deep Neural Network Classifier for Android Malwares Detection using Feature Combination

Saman Mirza Abdullah
Department of Software Engineering, Koya University, University Park,
Danielle Mitterr and Boulevard, 44018 Koya, Iraq
Saman.miza@koyauniversity.org

Abstract: Cases of Android security defence system penetration has increasedin recent years. This increase has prompted the investigation and evaluation of the execution behaviours of malicious applications in mobile systems, especially, Android Operating Systems (OSs). The main challenge of the detection systems is making many false alarms which are related to misclassifying the malicious behaviours as normal. This study proposes a new approach that combines the Android mobile features for targeting malwares and uses a deep feed forward neural network for detecting malicious activities or behaviours. The study employs different sets of mobile malware features to train and test the detection model. Results showed improved accuracy of detection rates when the detection model is trained with combined features.

Key words: Android mobile, malwares, deep neural classifiers, feature combination, feedforward, detection model

INTRODUCTION

Smart mobile devices are open, portable and popular platforms for using various applications. The number of downloaded mobile applications continuously increases annually. In fact, 197 billion applications were downloaded in 2017 and 290 billion in 2018. By 2021, the number of downloads is predicted to increase to 352.9 billion (Anonymous, 2018). This rapid increase is attributed to the convenience of information sharing, portable communication and various services that the mobile applications offer. Another factor is that these applications are free. Nevertheless, some mobile applications increase the vulnerability of smart mobile devices to threats and attackers (Kavitha, 2015).

Many attackers target mobile devices not only because they use several applications but also because attackers aim to breach privacy and subsequently, acquire personal information. Android devices which hold leading positions in the market have been targeted by 97% of total mobile malwares (Yan and Yan, 2018). Similarly, iOS devices have been exposed to serious attacks. These trends prompted extensive research in the field of mobile malwares, especially, those affecting Android devices.

Surveys on mobile malwares were performed by Yan and Yan (2018) Abdullah *et al*. (2018). Yan and Yan (2018) surveyed most researches that proposed dynamic tools. The researcher starts with defining the dynamic techniques proposed for mobile malware detection, identification of mobile malware features and review of the criteria and performance evaluation. The study argued that further tests that show the effectiveness of dynamic-based mobile malware detection systems must be further explored. Meanwhile Ricky and Gulo (2015) comprehensively investigated Android malware detection systems for mobile devices. Abdullah *et al*. (2018) compared different mobile malwares and several defence systems.

Most state-of-the-art systems classify detection systems into static and dynamic analysis systems. Static analysis systems aim to identify known malicious features or gaps in the code segments of suspected applications without execution. On the other hand, dynamic analysis systems focus on known malicious behaviours to classify applications at runtime. Both systems require the trade-off between the accuracy rate of detecting zero-day malwares and the efficiency of resource consuming as detecting and classifying mobile malwares cannot be performed perfectly without using the advantages of both systems (Kavitha, 2015; Naway and Li, 2019). The reasons are related to the security requirements of mobile defence systems. One of these requirements is detecting zero-day malwares and intrusion with low false alarm rates (preferred zero rate).

Corresponding Author: Saman Mirza Abdullah, Department of Computer Engineering, Ishik University, 44001 Erbil, Iraq
saman.mirza@ishik.edu.iq

Another security issue associated with mobile malwares is related to the increase in Android-based OSs within the era of Internet of Things (IoT). The number of Android devices connected to the internet continuously increases. One of the great characteristics of these devices is direct communication between devices and this characteristic is ideal for malware propagation (Naway and Li, 2019). In this regard, the present study proposes a method that combines two different sets of factors: the calls and permissions requested by a suspected mobile application and physical status of mobile hardware that the same application may activate or deactivate. The present combination approach should minimize the false alarm rate during the detection of malwares in Android mobile devices.

Work contributions: The performance indicators used in most mobile malware classifier models can be used in the evaluation of false alarm rate. Decrease in false alarm rate indicates improvement in detection models. This study aims to decrease the false alarm rates in Android mobile malware detection systems. Artificial classifier models are tested and trained with the combination of two sets of features that represent mobile malware behaviours. These models can be applied to the static and dynamic analysis systems for mobile malwares. Furthermore, a Deep Neural Network (DNN) is used in mobile detection. DNNs are often used in text, speech and image classification or clustering. However, DNN is very rarely used in security filed, especially, in the Android malware detection systems.

Literature review: Resrarch by Kavitha (2015) suggests that Android malware detection methods for mobile devices are still immature. Most researcher are using either static analysis or dynamic analysis. Novelties in both systems are presented in this study.

Static analysis detection: Methods that utilize static analysis are known to be quick, inexpensive and accurate (i.e., depending on scanning a suspicious mobile application without execution). Mobile malware analysts perform preliminary checks on a suspicious applications by using static analysis and check the availability of known malicious characteristics (Yan and Yan, 2018). Many features in mobile applications can be extracted and checked statically. The first feature of the Android mobile applications is the usage of Application Programming Interface (API) functions. Information about API usage such as type and frequently usage can be extracted through resources inside a package file (e.g., classes.dex, resources.arsc and AndroidManifest.xml. Several novel works on API usage information as a feature for detecting mobile malwares and malicious activities have been conducted. This information was incorporated to many statistical tools and machine learning algorithms (Alazab *et al*., 2011; Qiao *et al*., 2014; Ghani *et al*., 2015). Another characteristic that can be studied and analysed statically is considering the security risks that may occur from request permissions such as requests that control other applications or some hardware parts of the mobile devices. Android mobile applications send permission codes to a system individually or collaboratively as a group or requests permission from another application. Analysing the type of permission can defeat malicious characteristics inside a segment of code (Aung and Zaw, 2013). Finding the similarity between a known malicious permission and a suspicious permission is the result of detection (Feizollah *et al*., 2015). Disassembling codes is another method of static analysis for mobile malware detection. The process, called reverse engineering is performed after code segmentation. However, the flow of the code execution is still monitored (Yan and Yan, 2018; Daniel, 2013). The great gap or disadvantage for this analysis method is detecting zero-day malwares or system penetration events that occur at the time of download or execution of an Android application which always come with high false alarm rate.

Dynamic analysis detection: Dynamic analysis, unlike static analysis, depends on the monitored execution behaviours of mobile applications. Execution and monitoring are normally performed in a controlled environment, so that, malicious code injections are prevented (Bhatia and Kaushal, 2017; Ayasrah, 2019). The process monitors the features of the execution behaviours for tracking and identifying predefined taint behaviours. Several works monitored the drawback of battery efficiency and the power consumption of a suspected application (Bhatia and Kaushal, 2017); they also examined the network traffic of the application during execution. The work depended on some packet-related features (e.g., the IP address of source and destination, the port number of the source and destination, the start and end times of the network connection and the flow direction of the connectivity). The most common dynamic monitoring systems use the behaviours of system calls; API usage and the flow of sensitive data (Bhatia and Kaushal, 2017).

Deep learning-based detection: Several studies that used deep learning methods for Android malware detection differ in the number of input features used and in the algorithm used for training. Zhang *et al*. (2018) classified

Android malwares through three main steps, namely, feature extraction, feature embedding and deep learning-based detection. In the last step, malware detection was performed with convolutional neural network. Xu *et al.* (2018) proposed the use of Deep Neural Network (DNN) named as DeedRefiner, as an Android malware detection system to increase the speed of malware feature extraction in order to keep up with the speed of Android malware evolution. Meanwhile, a multimodal DNN that includes most of the Android malware features in contrast to DeepRefiner was proposed by Kim *et al.* (2019) and Kundeti *et al.* (2019) who argued that collecting malware features from many Android devices (not only mobile device) and then building a good representative method on the basis of the similarity among features reduce complexity. Then, they evaluated the proposed approach in terms of accuracy and time complexity for updating the model. Their results indicated 9% improvement in their model when two features were used relative to when five features were used. The time complexity increased by 10%. Overall, they obtained 98% accuracy.

The present study follows a feature combination method and uses a DNN to improve the accuracy rate of Android malware detection while minimizing the number of utilized features in order to minimize complexity and maintain the accuracy rate.

MATERIALS AND METHODS

The proposed framework: Figure 1 framework of combined DNN for Android Malware Detection Model illustrates the framework proposed here. The study consists of two main parts: collecting samples and extracting features and using DNN and combination method to train the detection model and test its accuracy performance.

Collection of samples: A dataset with malicious and normal application vector representation is necessary to the training of the combination malware detection model. The benign samples from Google Play were collected from June 2018 July 2018. To exclude the dataset from any malicious activity, we tested the collected applications with VirusTotal. VirusTotal is a website that provides malware scanning services with more than 60 scanners. Malware applications were then collected from the Android Malware Genome Project and VirusShare (Wang *et al.*, 2017). A total of 10,600 Apk samples were collected, of which 5300 were malwares and 5300 were benign applications. The malware samples belong to 49 different families and the benign applications cover different categories (call and contacts,

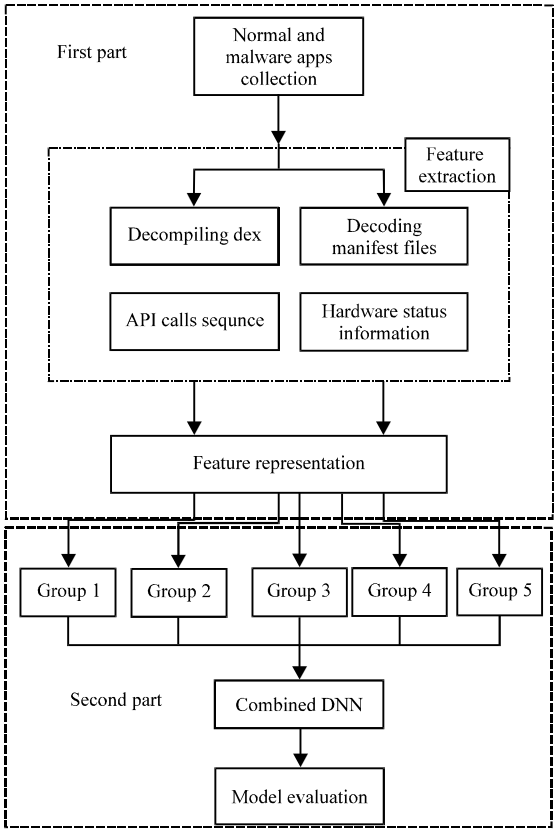

Fig. 1: Framework of combined DNN for android malware detection model

Table 1: Mathematical notion of dataset

Notions	Discretions
N_p	A set represents the execution patterns of normal Android apps (this may include permission request or API call sequence)
M_p	A set represents the execution patterns of malicious. Android apps (this may include permission request or API call sequence)
F_{np}	A set of physical properties of each pattern (element) in the set NP
F_{MP}	A set of physical properties of each pattern in the set MP

education, fun and games, GPS and travel, internet, lifestyle, news and weather, productivity, utilities, business, communication applications, entertainment, fitness and personalization). Different categories were used because Android applications are designed for many lifestyle applications and each category has a special way to call mobile hardware and other installed applications and different policies for misusing these calls. Figure 2 shows the number of malware families utilized by this research.

Table 1 presents the mathematical notation of the dataset. This notation describes the normal and malicious

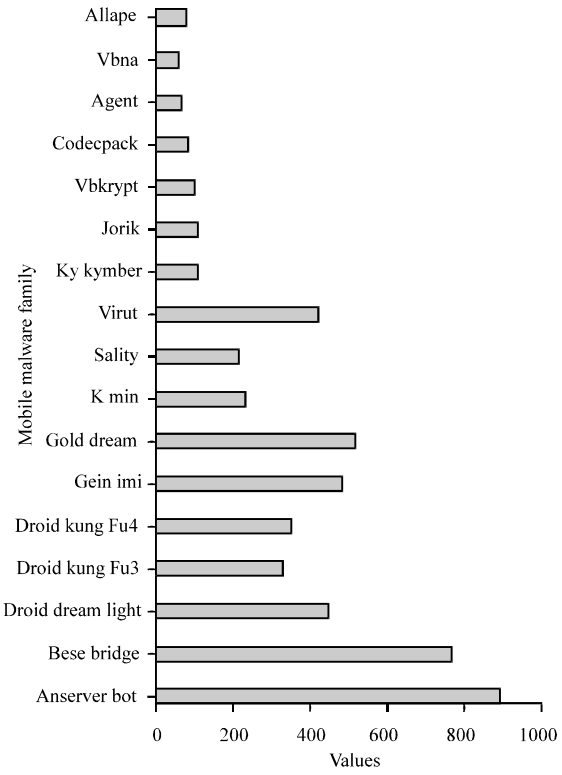

Fig. 2: Number of mobile malwares per malware's family

patterns that should be extracted from the known Android applications. Every element inside the FNP and FMP sets has a corresponding element in NP and MP (i.e., the ith element of FNP is the physical characteristic of the ith element in the NP and the same is true for FMP and MP). The main idea for the combination process is identifying the pattern of a suspicious application within Np or Mp and the physical properties of the suspected application should confirm that the first identification is true (i.e., if the suspected app is classified into Np, then the physical properties of that application should explain the same). The normal and malware pattern sets should be updated in case of any conflict and the model should be trained with the new set.

Feature extraction and representation: The idea of extracting features is to investigate which patterns are visible in the malware and benign applications. The challenge in statistical identification is the equal use of some permissions and API calls by the malware and benign applications. Such permissions include internet permission which is called by 4738 malwares and 4823 normal applications. Therefore, distinguishing applications according to the misuse of permissions decreases the reliability of the detection results. However, many types of permissions are called by malwares at higher rates than benign applications. For example, RESTART_PACKAG, WRITE_APN_SETTING, SEND_SMS, READ_SMS, RECEIVE_SMS, WRITE_CONTACTS and CHANGE_WIFI_SETTING are more frequently called by malwares. Therefore, such permissions can be used in accurately distinguishing clean applications from malicious ones. Malwares and normal applications have different calling sequences for APIs and normally such differences between the calling sequence of APIs can be used for distinguishing normal and malicious Android applications. API sequences [socket (), bind (), getsockname (), connect (), sendmsg ()] are more frequently used by normal applications than malwares. Therefore, any abnormality in the calling sequence in SMS activities can be considered malicious.

Calling some services related to the device and system is another security issue in Android devices. Many Android malwares use GetDeviceId () to obtain device numbers or Runtime.exec () to control devices and root them. Such requests are rarely encountered during the execution of a normal application. Other patterns of calling and requesting services can be used in circumventing malicious activities (combining two requests together). Many malwares that access private information check user location through the APIs (ACCESS_FINE_LOCATION) and (SEND_SMS). Clear patterns for normal applications and malwares can be revealed through all these features and from 10,600 samples. The next step of our approach is to represent features with Boolean expression. Each collected application will be converted to v vector and each vector will be labeled as (0), if it is a normal application or (1) if it is a malware. For example, if a suspicious application requests permission to send a message through (SEND SMS) while the status of screen is on (android.permission.WAKE_LOCK) then this application will be labeled as (1) because no user can send an SMS while the screen of the mobile is off. Five types of datasets were built according to the extracted features. The first dataset is the combination between Np and Mp with reference to the permission requests. The second is the combination between Np and Mp with reference to the sequence of the API calls. The third is the combination between Np, Mp, FNP and FMP with reference to permissions and the fourth is the same combination but with reference to the API calls. The last one is the combination of all dataset with the present of all features.

$$input\ set(v) = \begin{pmatrix} 0 \\ 0 \\ 1 \\ \dots \\ \dots \\ \dots \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

Fig. 3: A sample of the input dataset

Table 2: Size of vector (v) in each dataset's group

Group No.	Features	No. of bits
1	Only permissions	256
2	Only API calls sequences	256
3	Permissions+Physical properties	512
4	API sequence+Physical properties	512
5	(Permissions+Physical properties)+ (API sequence+Physical properties)	1024

Figure 3 shows a sample of the input dataset used in this research. The length of the constricted vector (v) is changed according to the group number. Table 2 shows the vector (v) size in each group of the dataset.

Deep neural network detection model: The goal of a Deep Feedforward Neural Network (DFNN) is to estimate function f^* in order to classify malicious applications from clean ones in Android mobile devices. The classifying function $y = f^*(x)$ maps an input x to category y which is defined as $f^*(x; \bullet)$ where $\bullet$ is the result of the mapping process. The structure of any DFNN consists of some l layers (input, hidden and output), each of which has several interconnected units (nodes). This study proposes four layers l = (0, 1, 2, 3) where input values of the layer y^l is from y^{l-1}. The number of nodes at the input layer is changed according to the size of $\bullet$ which is the set defined as input dataset in section 4.2. A total of 1024 nodes are in the hidden layer which provides the optimum approximation. Parametric Rectified Leaner Unit (PReLU) is the activate function employed for nodes in the hidden layer. The PReLU activate function considerably increases the speed of network training over the traditional sigmoid function (Krizhevsky *et al* 2012; Sakthivel *et al*., 2019). Thus, using PReLU function is preferred than sigmoid in the nodes of the hidden layers. The transform function at each unit which maps input x to a computable output could be illustrated as in Eq. 1:

$$y^l = F\left(W^l.y^{l-1} + b_l\right) \quad (1)$$

Where:

W^l = The associated weight's value with the layer l

b_l = Denotes the associated biase's value with the layer l

All nodes at the hidden layers (1, 2) will perform computation based on the function shown in Eq. 1. However, the following function (sigmoid) is used for the nodes laid in the output layer. Finally, the following indicators (Rashid and Abdullah, 2018; Rashid *et al*., 2016) are evaluated:

- True Positives (TP): this index measures the number of successfully classified malicious applications
- False Negatives (FN): this index measures the number of misclassified malicious applications
- False Positives (FP): this index measures the number of benign applications distinguished correctly from malicious group
- Some important indicators also utilized by this study are false-positive rate (FPR = FP/FP+TP), false-negative rate (FNR = FN/FN+TP) and accuracy (ACC = TP+TN/P+N)
- Precision (P): the percentage of positive prediction (i.e., the percentage of the detected malware out of all sample applications): P = TP/TP+FP
- Area Under the Curve (AUC): this parameter shows how the time complexity of the model is changing over the change of feature dimensionality: ACC = TP+TN/TP+TN+FP+FN and thus, crucial to case detection
- Recall (R = TP/TP+FN): the percentage of correct classification of malicious behaviors among the scanned malware samples
- F1 score (F1 = 2×P×R/P+R): the measure that combines precision and recall

RESULTS AND DISCUSSION

The evaluation step starts with a cross-validation experiment which have been done by many works that utilized deep learning neural network (Rosyda *et al*., 2019; Choi, 2019). The study depends on k-fold validation method with (k = 2, 3, 5 and 10). Accordingly, the dataset is partitioned into five parts where four parts were used for training and the other (20%) for testing.

The dataset has a dimension size of m×n where, m represents the number of samples used in this study (10,600). Meanwhile, n changes within the change in the groups of the dataset (256, 512 and 1024). For the cross-validation, n = 1024 and the tested DNN structure will be 1024 nodes at input layer and two hidden layers with 150 nodes at each. Table 3 shows the summary of the

cross-validation process. The results show significant similarity among the dataset portions. The training phase of DFNN is applied individually to each group of the dataset. The subsets have different numbers of attributes, however, the sub-datasets have the same number of samples (9600). The rate of AUC at each epoch of training is recorded for the measurement training process performance. Fig. 4 shows the relationship between AUC rate and epoch in each feature subset. The figure shows that the process of training for all the dataset subgroups are smooth and no sharp gradient appears in all cases. However, the feature combination affects the accuracy of the (AUC) rate. The highest rate obtained is 98.78% when all features have been combined in the trained dataset. The lowest rate (84.11%) is obtained when the sequence of the API calls is considered as the only feature for detection because malwares often attempt to make their call behaviors similar to those of normal applications as possible (Fig. 5).

A total of 1000 randomly selected individuals from each sub-dataset is used for model testing. All performance indicators mentioned in section 4.3 are employed in the testing process.

Table 3: k-fold cross-validation testing

Folds	F1 (%)	P (%)	R (%)
2	96.921	97.121	96.982
3	97.631	97.778	96.896
5	97.748	97.867	97.687
10	98.091	98.208	98.139

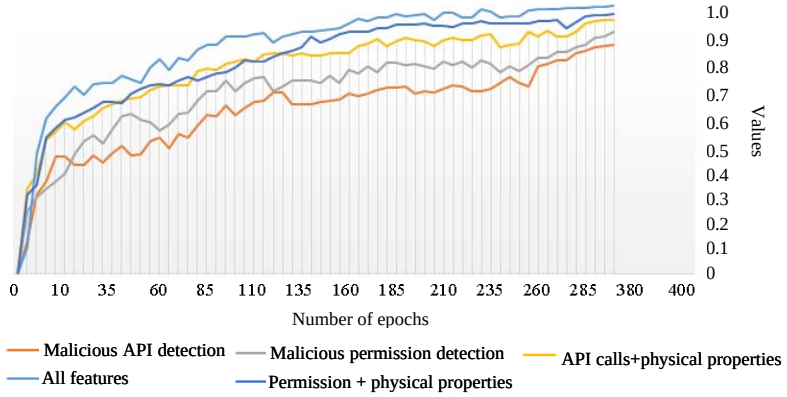

Fig. 4: Training process for individual sub-datasets

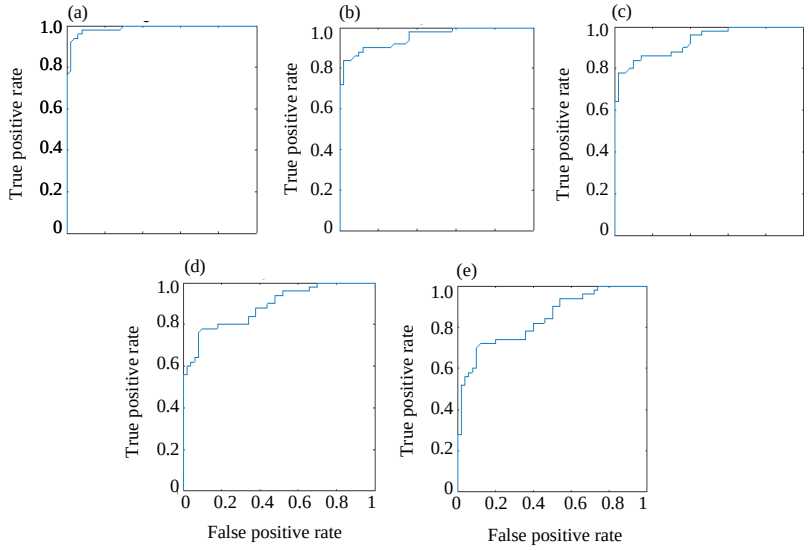

Fig. 5: a-e) AUC rate of detection model with feature combination

Figure 5 shows the AUC rate of the detection model in each sub-dataset in the form of TPR-FPR relationship. Figure 5a shows the relation between the TRP and FPR for our detection model that is trained with all combination features. The obtained AUC rate is 98.87%. Figure 5b shows the AUC of the model when trained with two feature combinations (permission request and physical hardware accessing features). The rate is 95.34%. Figure 5c shows the AUC rate for the API calls and physical properties features (i.e., 93.08%). Figure 5d shows the permission request feature with 88.87% rate and Fig. 5e shows the API call feature with 84.10% rate. The sub-figures highlight the impact of feature combination of Android applications on increasing the accuracy rate of detection or classification of Android mobile malwares.

CONCLUSION

Most Android malware detection models depend on the features of applications to distinguish malwares from benign applications. This study combines two interrelated features. The study showed that the combination of different features has impact on getting better accuracy rate of malware detection and classification.

A total of 10,600 samples are collected and divided equally between malware and benign applications. Five groups of input datasets are prepared and each group represents a unique combination of features. The collected dataset is validated. Then, a DFNN is structured, trained and tested. Most mobile malwares interact differently with system software, hardware and other installed applications. Malwares can be different from benign applications in many aspects, although, similar to benign applications.

ACKNOWLEDGEMENT

The researcher of this study would like to thank both universities (Koya and Ishik University) as they funded this project. Moreover, the research would like to thank Mr. Peshawa Mohammed Jamal (head of Software Engineering in Koya University) and Mr. Safwan Mawlood (head of Computer Engineering in Ishik University) for their valuable supports

REFERENCES

Abdullah, S.M., B. Ahmed and M.M. Ameen, 2018. A new taxonomy of mobile banking threats, attacks and user vulnerabilities. Proceedings of the 4th International Engineering Conference on Developments in Civil & Computer Engineering Applications (IEC2018), February 26-27, 2018, Erbil, Iraqi Kurdistan, pp: 372-384.

Alazab, M., S. Venkatraman, P. Watters and M. Alazab, 2011. Zero-day malware detection based on supervised learning algorithms of API call signatures. Proceedings of the 9th Australasian Conference on Data Mining (AusDM'11) Vol. 121, December 01-02, 2011, Australian Computer Society Inc., Ballarat, Australia, ISBN:978-1-921770-02-9, pp: 171-182.

Anonymous, 2018. McAfee labs threats report. MaCafee, Santa Clara, California, USA. https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-jun-2018.pdf

Aung, Z. and W. Zaw, 2013. Permission-based android malware detection. Intl. J. Sci. Technol. Res., 2: 228-234.

Ayasrah, F.T.M., 2019. Extension of Technology Adoption Models (TAM, TAM3, UTAUT2) with trust; Mobile learning in Jordanian Universities. J. Eng. Appl. Sci., 14: 6836-6842.

Bhatia, T. and R. Kaushal, 2017. Malware detection in android based on dynamic analysis. Proceedings of the 2017 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), June 19-20, 2017, IEEE, London, UK., ISBN:978-1-5090-5064-2, pp: 1-6.

Choi, Y., 2019. Weaknesses of anonymous mutual authentication schemes for roaming service with smart cards. J. Eng. Appl. Sci., 14: 7013-7019.

Daniel, G., 2013. Principles of Artificial Neural Networks. 3rd Edn./Vol. 7, World Scientific, Singapore, ISBN:978-981-4522-73-1, Pages: 384.

Feizollah, A., N.B. Anuar, R. Salleh and A.W.A. Wahab, 2015. A review on feature selection in mobile malware detection. Digital Invest., 13: 22-37.

Ghani, S.M.A., M.F. Abdollah, R. Yusof and M.Z. Mas'ud, 2015. Recognizing API features for malware detection using static analysis. J. Wirel. Netw. Commun., 5: 6-12.

Kavitha, K., 2015. Mobile banking supervising system-issues, challenges and suggestions to improve mobile banking services. Adv. Comput. Sci. Intl. J., 4: 65-67.

Kim, T., B. Kang, M. Rho, S. Sezer and E.G. Im, 2018. A multimodal deep learning method for android malware Detection using various features. IEEE. Trans. Inf. Forensics Secur., 14: 773-788.

Krizhevsky, A., I. Sutskever and G.E. Hinton, 2012. Image net classification with deep convolutional neural networks. Proc. Neural Inf. Process. Syst., 1: 1097-1105.

Kundeti, N.P. and M.V.P.C.S. Rao, 2019. A combined approach for privacy preserving classification mining. J. Eng. Appl. Sci., 14: 188-194.

Naway, A. and Y. Li, 2019. Android malware detection using autoencoder. Intl. J. Comput. Eng. Appl., 12: 1-9.

Qiao, Y., Y. Yang, J. He, C. Tang and Z. Liu, 2014. CBM: Free, automatic malware analysis framework using API call sequences. Proceedings of the 7th International Conference on Knowledge Engineering and Management (ISKE 2012), December 01, 2012, Springer, Beijing, China, ISBN:978-3-642-37831-7, pp: 225-236.

Rashid, A., S.M. Abdulla and R.M. Abdulla, 2016. Decision support system for diabetes mellitus through machine learning techniques. Int. J. Adv. Comput. Sci. Appl., 7: 170-178.

Rashid, T.A. and S.A. Abdulla, 2018. A hybrid of artificial bee colony, genetic algorithm and neural network for diabetic mellitus diagnosing. ARO-Sci. J. Koya Univ., 6: 55-64.

Ricky, M.Y. and R.S. Gulo, 2015. A comprehensive study for mobile android malware detection. Proceedings of the 2015 International Conference on Network Security & Computer Science (ICNSCS-15), June 10-11, 2015, Antalya, Turkey, pp: 13-17.

Rosyda, S.A.S., B. Irawan and A.L. Prasasti, 2019. Design of Arabic recognition application using convolutional neural network. J. Eng. Appl. Sci., 14: 6982-6990.

Sakthivel, N.K., N.P. Gopalan and S. Subasree, 2019. Deep learning based human diseases pattern prediction technique for high dimensional human diseases data sets. Intl. J. Eng. Res. Tech., 12: 204-211.

Wang, S., Z. Chen, X. Li, L. Wang and K. Ji *et al*., 2017. Android malware clustering analysis on network-level behavior. Proceedings of the International Conference on Intelligent Computing, August 7-10, 2017, Springer, Liverpool, England, UK., ISBN:978-3-319-63308-4, pp: 796-807.

Xu, K., Y. Li, R.H. Deng and K. Chen, 2018. DeepRefiner: Multi-layer android malware detection system applying deep neural networks. Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS&P), April 24-26, 2018, IEEE, London, UK., ISBN:978-1-5386-4229-0, pp: 473-487.

Yan, P. and Z. Yan, 2018. A survey on dynamic mobile malware detection. Software Qual. J., 26: 891-919.

Zhang, Y., Y. Yang and X. Wang, 2018. A novel android malware detection approach based on convolutional neural network. Proceedings of the 2nd International Conference on Cryptography, Security and Privacy (ICCSP 2018), March 16-19, 2018, ACM, Guiyang, China, ISBN:978-1-4503-6361-7, pp: 144-149.