Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2017) 25: 2232 – 2242
© TÜBİTAK
doi:10.3906/elk-1602-107

Android malware classification based on ANFIS with fuzzy c-means clustering using significant application permissions

Altyeb ALTAHER[*], **Omar BARUKAB**
Department of Information Technology, Faculty of Computing and Information Technology-Rabigh,
King Abdulaziz University, Rabigh, Saudi Arabia

Received: 08.02.2016 • **Accepted/Published Online:** 29.08.2016 • **Final Version:** 29.05.2017

Abstract: Mobile phones have become an essential part of our lives because we depend on them to perform many tasks, and they contain personal and important information. The continuous growth in the number of Android mobile applications resulted in an increase in the number of malware applications, which are real threats and can cause great losses. There is an urgent need for efficient and effective Android malware detection techniques. In this paper, we present an adaptive neuro-fuzzy inference system with fuzzy c-means clustering (FCM-ANFIS) for Android malware classification. The proposed approach utilizes the FCM clustering method to determine the optimum number of clusters and cluster centers, which improves the classification accuracy of the ANFIS. The most significant permissions used in the Android application selected by the information gain algorithm are used as input to the proposed approach (FCM-ANFIS) to classify applications as either malware or benign applications. The experimental results show that the proposed approach (FCM-ANFIS) achieves the highest classification accuracy of 91%, with lowest false positive and false negative rates of 0.5% and 0.4%, respectively.

Key words: Android malware, fuzzy c-means clustering, adaptive neuro-fuzzy inference

1. Introduction

The number of smartphone users and applications is increasing continuously. Android is the most popular smartphone operating system, based on data from the International Data Corporation [1]. Android dominated the market with a 78.0% share in the first quarter of 2015. The popularity of Android as a smartphone operating system has made it a target for malware writers and hackers. According to Kaspersky, about 200,000 unique samples of mobile malware were found in January 2014, and the number of accumulated malwares was up 34% from November 2013 [2]. Moreover, Sophos has accumulated 650,000 unique Android malware applications to date, with 2000 new benign Android malware applications detected every day [3].

Google introduced Bouncer to fight the fast spread of Android malware applications by checking the applications submitted to the Google Play Store for malware. According to Google reports, Bouncer has managed to decrease the number of malware in the Play Store by 40% since its utilization in February 2012 [4]. F-Secure noted that other markets accommodate 5%–8% malicious applications [5].

Because Android is the most dominant smartphone operating system, malware detection solutions are needed urgently. Our work presents an adaptive neuro-fuzzy inference system with fuzzy c-means cluster-

[*]Correspondence: altypaltaher@gmail.com

ing (FCM-ANFIS) for Android malware classification, which achieves higher accuracy than the classification methods previously reported.

Several classification methods have been proposed to distinguish between malware applications and benign ones. The classification decisions depend on analyzing the application's static [6] or dynamic [7] behavioral features. Dynamic features are obtained by monitoring the behaviors of the application during the run-time while static features are obtained by decompiling the Dalvik byte code of the Android applications. The classification accuracy depends on the features quality and the effectiveness of the classifier. Android malware classification is an active research area and more effort is needed to improve the classification accuracy of malware applications [8]. The state-of-the-art Android malware detection approaches give different values for false positive and false negative rates. For example, AndroidLeaks [9] and the approach used in [6] give false positive rates of 35% and 15%, respectively. K-ANFIS [8] and Andromaly [10] give false positive rates of 10%. DREBIN [11] gives a better false positive rate of 6%. The approach proposed in this paper (FCM-ANFIS) aims to improve the Android malware classification accuracy. The contributions of our research are:

- Measuring the most significant permissions to discriminate optimally between malware and goodware apps.
- Integrating FCM-ANFIS for Android malware classification and improving the learning process and detection accuracy.
- Investigating the potential of neuro-fuzzy inference systems in classifying Android malwares.

This paper is organized as follows: Section 2 reviews the previous work on Android malware detection. Section 3 presents FCM-ANFIS. The proposed approach (FCM-ANFIS) for Android malware classification is introduced in Section 4. Section 5 presents an experimental evaluation of the proposed approach compared with other approaches. Finally, Section 6 concludes the paper.

2. Related work

To mitigate the rapid spread of Android malware, there is a real need for efficient and effective approaches to malware detection. Two dominant approaches for detecting Android malware are behavior-based and signature-based.

2.1. Behavior-based malware detection approaches

Behavior-based malware detection approaches detect malware by dynamic analysis of different features and attributes that help to distinguish anomalous behaviors. The features that could be monitored and traced by these approaches include the permissions and the system calls used by the application, network traffic activities, and user logs. Behavior-based approaches, such as those in [12] and [13], are not dependent on a specific pattern of codes, but they try to learn the behavior of the normal applications to detect the suspicious behavior, which is significantly different from the behavior of the benign applications.

2.2. Signature-based malware detection approach

A signature-based malware detection approach is used in [14] to classify a program as malware if the source code or binaries include a set of instructions or commands that are matched by a suspected pattern. Since malware writers can develop obfuscation techniques, signature-based approaches can be easily evaded [15]. Thus, these malware signatures need to be continuously updated when a new version of the same malware family appears.

Machine learning techniques have been used successfully in many applications, and this is promising for distinguishing between Android malware applications and benign applications in order to protect user devices and application markets from malware applications [16–20]. DREBIN [11] statically extracted the API calls and permissions as features and then applied a support vector machine for the classification of Android applications as either malware or benign ones.

DroidMiner [21] used a behavioral graph to represent the API calls and the connections between the API calls in the Android application, and then applied machine learning techniques to discriminate between the malware and benign applications. DroidSIFT [22] used the weighted API dependency graph to classify Android applications by examining the graphs for similarity to known graphs of benign applications to find the malware applications. Crowdroid [23] extracted the system calls from the Android application and classified the applications as goodware or malware using the K-means clustering algorithm.

3. Fuzzy c-means clustering algorithm and the adaptive neuro-fuzzy inference system

3.1. Fuzzy c-means clustering algorithm

The fuzzy c-means (FCM) algorithm is a well-known fuzzy clustering technique, which has been used in many clustering applications [24].One of its advantages is that it puts the data points in one of the predetermined clusters. FCM works by computing the degree of membership function, which describes how much the data point belongs to a certain cluster. The number of iterations performed by FCM to compute the degree of membership depends on the level of required accuracy. The FCM algorithm includes the following steps:

Step 1: The input data to be clustered are M-dimensional N data elements denoted by $xi(i = 1, 2, \dots, N)$.

Step 2: Consider the number of clusters to be determined as C, where 2 $\leq$ C $\leq$ N.

Step 3: Choose a suitable degree of cluster fuzziness f $>$ 1.

Step 4: Initialize the membership matrix U of size N $\times$ C $\times$ M randomly, such that

$$U_{jim} \in [0,\ 1] \sum_{j=1}^{c} U_{jim} = 1.0$$

$$\mathbf{\sum_{j=1}^{c} U_{ijm} = 1.0}$$

for each **i** and a fixed value of m.

Step 5: Find the cluster centers $\mathbf{CC_{jm}}$ for the **jth** cluster and its mth dimension by using Eq. (1).

$$\mathbf{CC_{jm}} = \frac{\sum_{i=1}^{N} \mathbf{U_{ijm}^{f} x_{im}}}{\sum_{i=1}^{N} \mathbf{U_{ijm}^{f}}}) \tag{1}$$

Step 6: Determine the Euclidean distance between the **jth** cluster center and the ith data element considering the mth dimension as in Eq. (2):

$$\mathbf{D_{ijm}} = \|(\mathbf{x_{im}} - \mathbf{CC_{jm}})\|\,. \tag{2}$$

Step 7: Use the $\mathbf{D_{ijm}}$ value to update the fuzzy membership matrix U. If $\mathbf{D_{ijm}} > 0$, then

$$\mathbf{U_{ijm}} = \frac{\mathbf{1}}{\sum_{\mathbf{c=1}}^{\mathbf{C}} \left(\frac{\mathbf{D_{ijm}}}{\mathbf{D_{icm}}}\right)^{\frac{\mathbf{2}}{\mathbf{f-1}}}}. \tag{3}$$

3.2. Adaptive neuro-fuzzy inference system (ANFIS)

The ANFIS system [25] combines a neural network with fuzzy logic techniques and utilizes both the intelligent abilities of the neural network and the advanced reasoning behavior of fuzzy logic.

A neural network is used in ANFIS to adjust and tune the parameters of the fuzzy system. The ultimate goal of the neuro-fuzzy approach is to adaptively enhance the performance of the fuzzy system using neural network methods. ANFIS employs the Takagi–Sugeno method to generate fuzzy "if-then" rules as in the following two rules:

Rule 1: If (x is $\mathbf{C_1}$) and (y is $\mathbf{D_1}$) then ($\mathbf{O_1} = \mathbf{m_{1x}} + \mathbf{n_{1y}} + \mathbf{z_1}$).

Rule 2: If (x is $\mathbf{C_2}$) and (y is $\mathbf{D_2}$) then ($\mathbf{O_2} = \mathbf{m_{2x}} + \mathbf{n_{2y}} + \mathbf{z_2}$).

Here, $\mathbf{x}$ and $\mathbf{y}$ represent the inputs, $\mathbf{C_i}$ and $\mathbf{D_i}$ are the fuzzy sets, $\mathbf{O_i}$ are the outputs generated by the fuzzy rules, and $\mathbf{m_i}$, $\mathbf{n_i}$, and $\mathbf{z_i}$ are parameters specified during the training phase. The ANFIS architecture contains five layers, as shown in Figure 1.

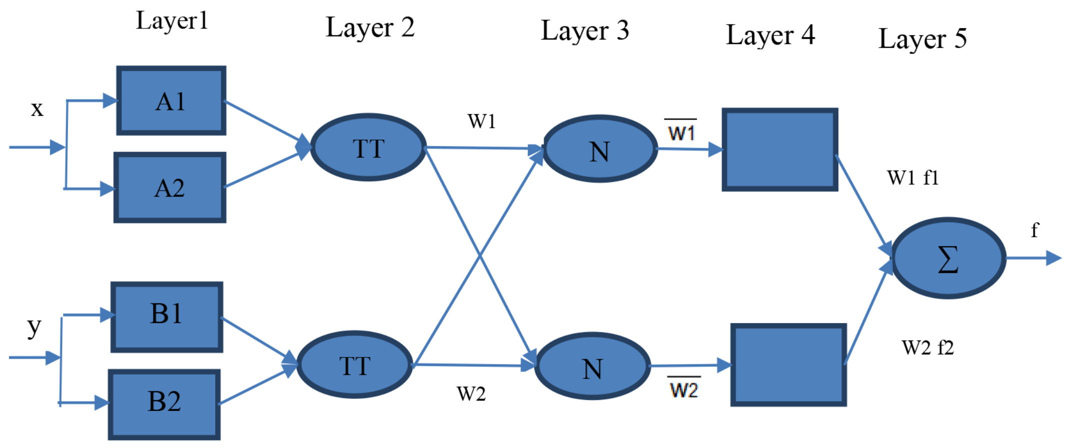


Figure 1. ANFIS architecture.

Layer 1: All the nodes in this layer are adaptive and produce the membership value based on the inputs and the applied membership function. The parameters in this layer are called premise parameters.

Layer 2: The firing strength of the rule is the output of each node in this layer.

Layer 3: The normalization of the firing strengths is implemented in this layer.

Layer 4: The output of all the nodes in this layer is the product of the first-order polynomial and the normalized firing strength.

Layer 5: A single node performs the calculation for the overall output of ANFIS as the summation of all incoming signals from the fourth layer.

4. The proposed Android malware classification approach

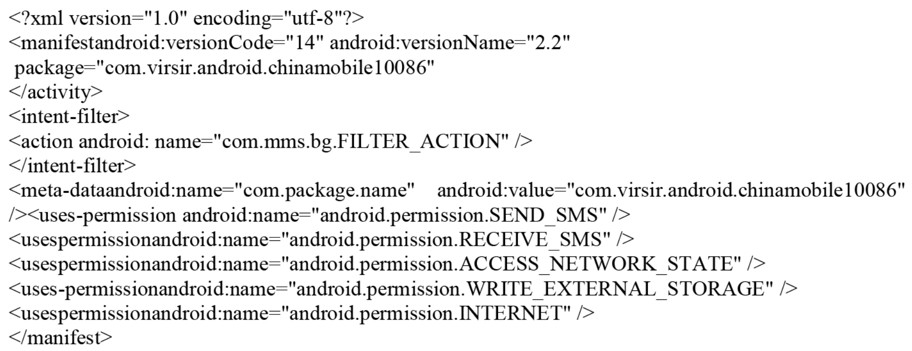
The proposed Android malware classification approach mainly consists of three phases. In the first phase, the features are extracted from the Android APK files and then the important and more significant features are selected using the information gain algorithm to improve the accuracy of classification. In the second phase, the FCM clustering algorithm is used to cluster the selected features into two main classes: malware and goodware. ANFIS is utilized to develop the classification model to differentiate between malware and goodware applications in the third phase.

4.1. Features extraction and selection

The Android application package (APK) consists of application resources, libraries, and an AndroidManifest.xml file [26], which contains the permissions needed by the API calls used by the Android application. Permissions are used by the Android operating system to control and restrict the access of the applications to the resources of the mobile devices; the device resources include the external storage, Internet access, and sending and receiving of SMS messages. Thus, in order to use the application properly, it is necessary to allow all the permissions needed by the application during its installation in the mobile device [4,8].

Google classifies Android permissions into four groups based on their level of importance. The first group is “normal”, a low-risk permission that provides the requesting application access to safe application-level resources. The second group is “dangerous”, a higher-risk permission that enables the requesting application to gain access to important user data or misuse of the mobile device resources and causes potential damage. The third type of permission is called “signature”, which is granted by the Android system only when the requesting application is signed using the correct certificate. The fourth type of permission, “signature or system”, is used where multiple developers have applications in the system’s image and want to share the resources [27].

Android introduced the permission system as a significant security mechanism. Therefore, the permissions requested by an application are the most used features to analyze the Android application [28]. Figure 2 shows a sample AndroidManifest.xml file for the Android malware application called *bgserv*, which uses the permission Android.permission.INTERNET to obtain access to the Internet, Android.permission.ACCESS_NETWORK_STATE to determine the network state and, Android.permission.SEND_SMS and Android.permission.RECEIVE_SMS to receive and send the user’s SMS messages.



```
<?xml version="1.0" encoding="utf-8"?>
<manifestandroid:versionCode="14" android:versionName="2.2"
 package="com.virsir.android.chinamobile10086"
</activity>
<intent-filter>
<action android: name="com.mms.bg.FILTER_ACTION" />
</intent-filter>
<meta-dataandroid:name="com.package.name"    android:value="com.virsir.android.chinamobile10086"
/><uses-permission android:name="android.permission.SEND_SMS" />
<usespermissionandroid:name="android.permission.RECEIVE_SMS" />
<usespermissionandroid:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permissionandroid:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<usespermissionandroid:name="android.permission.INTERNET" />
</manifest>
```

Figure 2. Snapshot of the AndroidManifest.xml file used by the Android malware application bgserv.

After extracting the permission features from the Android malware applications, the gain ratio algorithm

[29] is used to find the important application's features.

$$\mathbf{gain\_r}(\mathbf{XY}) = \frac{\mathbf{gain}(\mathbf{X},\mathbf{Y})}{\mathbf{split\_info}(\mathbf{Y})}, \tag{4}$$

$$\mathbf{split\_info}(\mathbf{Y}) = \sum\nolimits_{\mathbf{i}} \left(\frac{|\mathbf{Y_i}|}{|\mathbf{Y}|}\right) \log \frac{|\mathbf{Y_i}|}{\mathbf{Y}}, \tag{5}$$

where *gain_r* (X,Y) is the gain ratio of the feature X occurrence in class $\mathbf{Y}$. $\mathbf{Y_i}$ and $|\mathrm{Y}_i|$ denote the occurrence of features X in class $\mathbf{Y_i}$, the ith subclass of $\mathbf{Y}$, and the number of features in $\mathbf{Y_i}$, respectively.

Figure 3 shows the selected and ranked permissions. It is clear that the feature WRITE_APN_SETTING is the most significant feature in the differentiation between goodware and malware applications. The proposed approach uses the top 24 ranked features as inputs to the classifier.

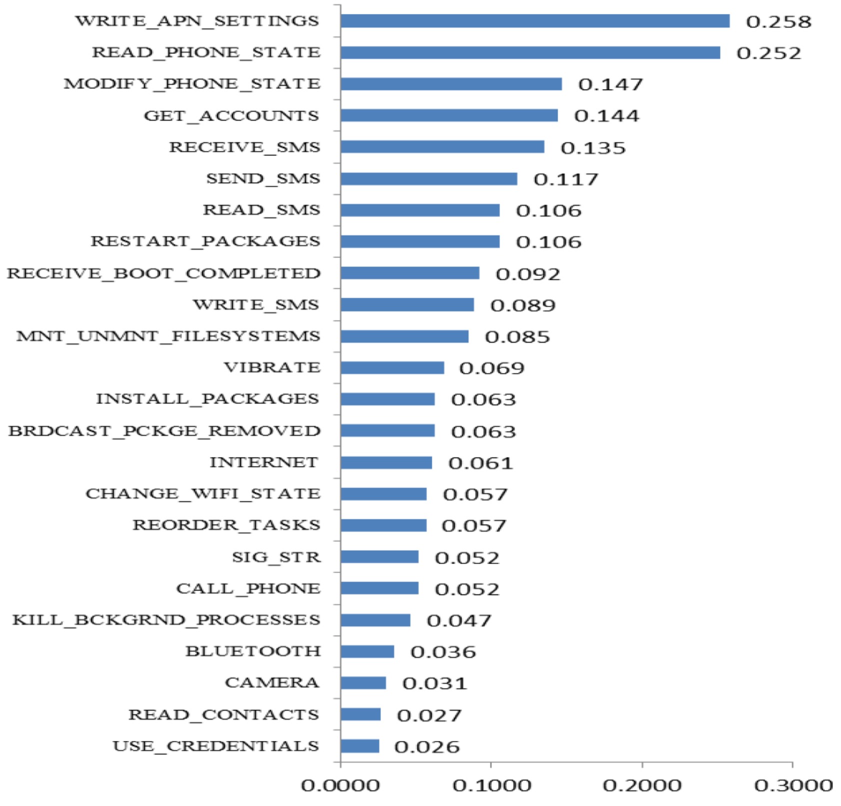


Figure 3. The selected features.

4.2. Classification of Android applications using the adaptive neuro-fuzzy inference system with fuzzy c-means clustering

The selected features of Android applications are grouped into clusters using the FCM clustering method. The aim of clustering is to group the similar features of Android applications into clusters to improve the accuracy of malware detection. The FCM clustering method is used to find the optimum number of clusters and cluster centers to improve the learning process and classification accuracy of ANFIS. The clustered features are used by ANFIS to train the Sugeno-type FIS by applying a hybrid learning algorithm to identify membership function parameters. The identification of the membership functions and their parameters is the most important phase in developing the neuro-fuzzy inference system [30]. We examined the performance of our approach, FCM-ANFIS, using four membership functions: Gaussian (gauss), sigmoid, generalized bell (*gbell*), and trapezoidal membership function (trap). The main goal of using these membership functions is to find which membership function is most suitable for our approach, i.e. curve, asymmetric, or straight line. We used the root mean square error (RMSE) to evaluate the performance of FCM-ANFIS for each membership function. The RMSE measures the difference between values implied by the membership function and the observed values:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(M_i - P_i)^2} \tag{6}$$

where n is the number of observations, M_i is the observed value, and P_i is the predicted value.

The membership functions types and parameters along with the achieved RMSE are shown in Table 1; the membership functions achieved different RMSE values. The highest RMSE value was achieved by the trap membership function with a value of 0.2943. The lowest RMSE value was achieved by the sigmoid membership function with a value of 0.1184. Since the sigmoid membership function achieved that result, it was adopted in the implementation of our approach, FCM-ANFIS.

Table 1. Performance of the proposed approach FCM-ANFIS using different membership functions.

MF	Number of parameters	Membership function type	RMSE
Gauss	3	Curve	0.1597
Sigmoid	2	Asymmetric and close curve (i.e. not open to the right or left)	0.1184
Gbell	2	Curve	0.2357
Trap	4	Straight lines	0.2943

5. Experimental classification results and analysis

To detect the Android malware applications, we used a dataset provided by the dataset available in the GNOME project [4] is used. Furthermore, we downloaded benign applications from the official Android application stores.

The standard tenfold cross-validation process is used in our experiments; the dataset is randomly distributed into 10 smaller subsets, and for training we used nine subsets and used one subset for testing. For every combination, we repeated the process 10 times.

To evaluate the performance of the classifiers, we used the confusion matrix shown in Table 2. A malware application is misclassified as goodware app, and a goodware application is misclassified as a malware application.

Table 2. The used confusion matrix.

		Predicted class	
		Malware	Goodware
Actual Class	Malware	True positive	False negative
	Goodware	False positive	True negative

True positive ratio (TPR):

$$\mathbf{TPR} = \frac{\mathbf{TP}}{\mathbf{TP} + \mathbf{FN}},$$

where TP (true positive) denotes the number of malware applications classified correctly and FN (false negative) denotes the number of malware applications classified incorrectly as benign applications.

False positive ratio (FPR):

$$\mathbf{FPR} = \frac{\mathbf{FP}}{\mathbf{FP} + \mathbf{TN}},$$

where FP denotes the number of benign applications incorrectly classified as malware applications and TN denotes the number of benign applications that are classified correctly as malware applications.

Accuracy is defined as the summation of correct classifications divided by the number of all classified samples.

Accuracy (ACC) = number of correct classifications / total number of classifications.

$$\mathbf{ACC} = \frac{\mathbf{TP} + \mathbf{TN}}{\mathbf{TP} + \mathbf{TN} + \mathbf{FP} + \mathbf{FN}}$$

The performance of the proposed approach, FCM-ANFIS, is compared with three classifiers: k-ANFIS [8], ANFIS [25], and DENFIS [31].

Table 3 shows the obtained Android malware classification results. Clearly, the proposed FCM-ANFIS achieved the highest accuracy with approximately 91% samples correctly classified with minimum false positive classification rate of 0.05% and minimum false negative classification rate of 0.04%. The proposed approach achieved the highest accuracy because it utilizes the FCM clustering algorithm to optimally determine the number of clusters that will be used as input to ANFIS.

Table 3. Results of Android malware classification using the four classifiers.

Algorithm	Accuracy	FP rate	FN rate
FCM-ANFIS	91%	5%	4%
k-ANFIS	75%	10%	15%
ANFIS	70%	13%	17%
DENFIS	70%	14%	16%

k-ANFIS achieved classification accuracy of precision of 75%, while ANFIS and DENFIS achieved the same classification rate of 70%. However, the three approaches (k-ANFIS, ANFIS, and DENFIS) achieved a high false positive classification rate of 10%, 13%, and 14%, respectively, and high false negative classification rates of 15%, 17%, and 16%, respectively, as shown in Figure 4.

The performance of the proposed approach, FCM-ANFIS, is also compared with the state-of-the-art approaches for Android malware detection as shown in Table 4. The proposed FCM-ANFIS achieved the lowest false positive and false negative rates of 0.5% and 0.4%, respectively.

6. Conclusion

In this paper we proposed and evaluated an effective approach for Android malware classification. In particular, FCM-ANFIS was applied to Android malware classification. The Android permissions were extracted from the

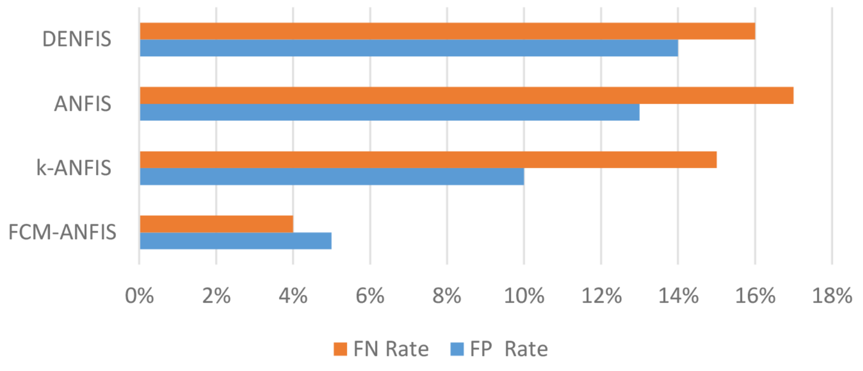


Figure 4. Results of the false negative and false positive rates obtained using the four classifiers.

Table 4. Performance comparison of the proposed approach, FCM-ANFIS, with the recent malware detection approaches.

Approach	Used features	Classification method	Used dataset	Classification accuracy
AndroidLeaks [9]	App permissions and API calls	Classification rules based on risky API calls	Dataset collected from various Android markets	FP = 35%
Crowdroid [23]	API calls	k-Means clustering algorithm	Dataset collected from Virus Total	FP = 20%
Dynamic detection [6]	Various CPU and memory measures	K-Means clustering algorithm, DT algorithm and naive bayes.	Dataset collected from Google Play and Virus Total.	FP = 15%
Andromaly [10]	Various CPU and memory measures	k-Means clustering, and naive bayes.	Dataset collected from Google Play	FP = 10%
RiskRanker [7]	Android system and application features	Several malware behavior signatures	Dataset collected from various Android markets	FN = 9%
DREBIN [11]	API calls and Permissions.	Support vector machine	Dataset collected from several apps markets.	FP = 6%
Our approach: FCM-ANFIS	Android permissions and API calls	FCM and ANFIS	Dataset collected from various Android markets	FP = 5% FN = 4%

applications and ranked based on their importance for classification using the information gain algorithm, and were used as inputs to our proposed approach, FCM-ANFIS, to classify the applications either as malware or benign. The evaluation of the proposed approach included the selection of the membership function and comparison with the other malware classification approaches. The experimental results in this paper showed the effectiveness of the proposed approach for the classification of Android applications as either malware or benign applications.

Acknowledgment

This work was supported by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, Saudi Arabia, under Grant No. 830-45-D1436. The authors therefore gratefully acknowledge the DSR technical and financial support.

References

[1] Yang HC, Chang WC. Ubiquitous smartphone platform for K-7 students learning geography in Taiwan. Multimed Tools Appl (in press).

[2] Yu S, Gu G, Barnawi A, Guo S, Stojmenovic I. Malware propagation in large-scale networks. IEEE T Knowl Data Eng 2015; 27: 170-179.

[3] Speed T, Nykamp D, Anderson J, Nampalli J, Heiser M. Mobile Security: How to Secure, Privatize, and Recover Your Devices. Birmingham, UK: Packt Publishing, 2013.

[4] Huang CY, Tsai YT, Hsu CH. Performance evaluation on permission-based detection for android malware. In: Advances in Intelligent Systems and Applications; 2013. Berlin, Germany: Springer, pp. 111-120.

[5] Rai PO. Android Application Security Essentials. Birmingham, UK: Packt Publishing, 2013.

[6] Amos B, Turner H, White J. Applying machine learning classifiers to dynamic android malware detection at scale. In: 2013 9th International Wireless Communications and Mobile Computing Conference; 1 July 2013; Cagliari, Italy. New York, NY, USA: IEEE. pp. 1666-1671.

[7] Grace M, Zhou Y, Zhang Q, Zou S, Jiang X. Riskranker: scalable and accurate zero-day android malware detection. In: Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services; 25 June 2012; Low Wood Bay, UK. New York, NY, USA: ACM. pp. 281-294.

[8] Abdulla S, Altaher A. Intelligent approach for Android malware detection. KSII Transactions on Internet and Information Systems 2015; 9: 2964-2983.

[9] Gibler C, Crussell J, Erickson J, Chen H. AndroidLeaks: automatically detecting potential privacy leaks in android applications on a large scale. In: International Conference on Trust and Trustworthy Computing; 13 June 2012; Vienna, Austria. Berlin, Germany: Springer. pp. 291-307.

[10] Shabtai A, Kanonov U, Elovici Y, Glezer C, Weiss Y. Andromaly: A behavioral malware detection framework for android devices J Intell Inf Syst 2012; 38: 161-190.

[11] Arp D, Spreitzenbarth M, Hubner M, Gascon H, Rieck K. DREBIN: Effective and explainable detection of android malware in your pocket. In: Network and Distributed System Security Symposium; 23 February 2014; San Diego, CA, USA.

[12] Fuchs AP, Chaudhuri A, Foster JS. SCAndroid: Automated Security Certification of Android Applications. Technical Report CS-TR-4991. College Park, MD, USA: Department of Computer Science, University of Maryland, 2009.

[13] Shabtai A, Moskovitch R, Elovici Y, Glezer C. Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey. Information Security Technical Report 2009; 14: 16-29.

[14] Xiong P, Wang X, Niu W, Zhu T, Li G. Android malware detection with contrasting permission patterns. China Commun 2014; 11: 1-14.

[15] Enck W, Gilbert P, Han S, Tendulkar V, Chun BG, Cox LP, Jung J, McDaniel P, Sheth AN. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. ACM T Comput Syst 2014; 32: 5.

[16] Rastogi V, Chen Y, Jiang X. Droidchameleon: evaluating android anti-malware against transformation attacks. In: 8th ACM SIGSAC Symposium on Information, Computer and Communications Security; 2013. New York, NY, USA: ACM. pp. 329-334.

[17] Cimpoe M, Anton D, Ciortuz L. Malware detection using machine learning. In: 2009 International Multiconference on Computer Science and Information Technology; 12–14 October 2009; Mragowo, Poland. New York, NY, USA: IEEE. pp. 735-741.

[18] Santos I, Nieves J, Bringas PG. Semi-supervised learning for unknown malware detection. In: International Symposium on Distributed Computing and Artificial Intelligence. Berlin, Germany: Springer, 2011. pp. 415-422.

[19] Firdausi I, Erwin A, Nugroho AS. Analysis of machine learning techniques used in behavior-based malware detection. In: 2010 Second International Conference on Advances in Computing, Control and Telecommunication Technologies; 2–3 December 2010; Jakarta, Indonesia. New York, NY, USA: IEEE. pp. 201-203.

[20] Zhou Y, Jiang X. Dissecting android malware: characterization and evolution. In: 2012 IEEE Symposium on Security and Privacy; 20–23 May 2012; San Francisco, CA, USA. New York, NY, USA: IEEE. pp. 95-109.

[21] Yang C, Xu Z, Gu G, Yegneswaran V, Porras P. Droidminer: Automated mining and characterization of fine-grained malicious behaviors in android applications. In: European Symposium on Research in Computer Security; 7 September 2014; Wroclaw, Poland. pp. 163-182.

[22] Zhang M, Duan Y, Yin H, Zhao Z. Semantics-aware android malware classification using weighted contextual API dependency graphs. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security; 3 November 2014; Scottsdale, AZ, USA. New York, NY, USA: ACM. pp. 1105-1116.

[23] Burguera I, Zurutuza U, Nadjm-Tehrani S. Crowdroid: behavior-based malware detection system for android. In: Proceedings of the 1st ACM workshop on Security and Privacy in Smartphones and Mobile Devices; 17 October 2011; Chicago, IL, USA. New York, NY, USA: ACM. pp. 15-26.

[24] Bezdek JC. Pattern Recognition with Fuzzy Objective Function Algorithms. Dordrecht, the Netherlands: Kluwer Academic Publishers, 1981.

[25] Jang JS. ANFIS: Adaptive-network-based fuzzy inference system. IEEE T Syst Man Cyb 1993; 23: 665-685.

[26] Santos I, Brezo F, Ugarte-Pedrero X, Bringas PG. Opcode sequences as representation of executables for data-mining-based unknown malware detection. Inform Sciences 2013; 231: 64-82.

[27] Jeon J, Micinski KK, Vaughan JA, Fogel A, Reddy N, Foster JS, Millstein T. Dr. Android and Mr. Hide: fine-grained permissions in android applications. In: Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices; 19 October 2012; Raleigh, NC, USA. New York, NY, USA: ACM. pp. 3-14.

[28] Feizollah A, Anuar NB, Salleh R, Wahab AW. A review on feature selection in mobile malware detection. Digit Invest 2015; 13: 22-37.

[29] Mori T. Information gain ratio as term weight: the case of summarization of IR results. In: Proceedings of the 19th International Conference on Computational Linguistics; 24 August 2002; Taipei, Taiwan. pp. 1-7.

[30] Singh R, Kainthola A, Singh TN. Estimation of elastic constant of rocks using an ANFIS approach. Appl Soft Comput 2012; 12: 40-45.

[31] Kasabov NK, Song Q. DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. IEEE T Fuzzy Syst, 2002; 10: 144-154.