BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **24**, No 3
Sofia • 2024 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2024-0024

ANFIS-AMAL: Android Malware Threat Assessment Using Ensemble of ANFIS and GWO

Nedal Nwasra[1], *Mohammad Daoud*[2], *Zahid Hussain Qaisar*[3]

[1]*University of Petra, Jorden*
[2]*American University of Madaba, Jorden*
[3]*Emerson University, Multan, Pakistan*
E-mails: nedal.nwasra@uop.edu.jo m.daoud@aum.edu.jo zahidhussainqaisar@gmail.com

***Abstract*:** *The Android malware has various features and capabilities. Various malware has distinctive characteristics. Ransomware threatens financial loss and system lockdown. This paper proposes a threat-assessing approach using the Grey Wolf Optimizer* (*GWO*) *to train and tune the Adaptive Neuro-Fuzzy Inference System* (*ANFIS*) *to categorize Android malware accurately. GWO improves efficiency and efficacy in ANFIS training and learning for Android malware feature selection and classification. Our approach categorizes Android malware as a high, moderate, or low hazard. The proposed approach qualitatively assesses risk based on critical features and threats. Our threat-assessing mechanism's scale categorizes Android malware. The proposed approach resolves the issue of overlapping features in different types of malware. Comparative results with other classifiers show that the ensemble of GWO is effective in the training and learning process of ANFIS and thus achieves 95% F-score, 94% specificity, and 94% accuracy. The ensemble makes fast learning possible and improves classification accuracy.*

***Keywords*:** *Malware, Ransomware, ANFIS, GWO, Android.*

1. Introduction

Various Android malware contains varying attack capabilities and has different features. The ransomware can lock systems and demand the user to pay ransom to get proper functionality or restore the system's previous state under ransom attack [1]. The botnets in the Android environment exploit resources to spread advertisements and other agendas. On the other hand, Adware uses the platform to throw advertisements and for marketing purposes [2, 3]. In some scenarios, the malware can have dual functionality as it may simultaneously contain adware and Ransomware capabilities [4, 5]. Therefore, the Android malware assessment is important for classification based on the capabilities.

Risk assessment and threat classification are effective in this measure regarding quality. Qualitative analysis is more suited to human-attributed analysis or natural language than analysis based on quantity. Android malware of different types

contains overlapping features and attributes [6, 7]. Due to overlapping properties, it is a daunting job to classify the Android malware quantitatively. Therefore, qualitative analysis with fuzzy sets is more suited to the scenario of Android malware classification. Unlike crisp sets, fuzzy sets can handle overlapping features. However, fuzzy inference systems are slow and suffer from low accuracy and precision in real-life applications [8, 9].

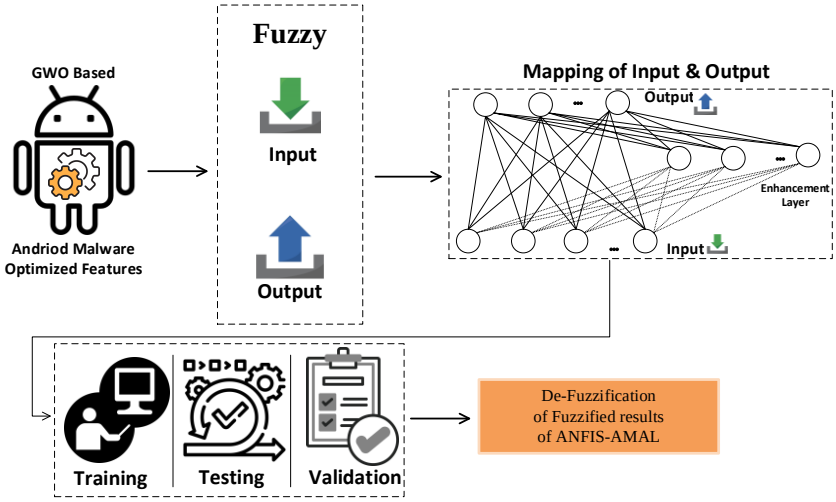

Fig. 1. Generic fuzzy inference system modelling

The fuzzy inference system is generally applied to problems containing overlapping properties and required output in qualitative value. Figure 1 shows the overall working of the fuzzy inference model. Generically, it represents input as crisp input passed to the fuzzifier module [10, 11]. The crisp input is converted into the fuzzy input and passed to the fuzzy inference system that provides output in fuzzy set output based on the knowledge base. The rule-based module of the knowledge base capitalizes on generating the appropriate output values in the last de-fuzzification inference unit [12, 13]. The figure above illustrates that the de-fuzzification inference unit de-fuzzifies the fuzzy output. The resultant output is crisp output when it is required. The proposed approach capitalizes the fuzzy inference system to categorize the Android malware.

Learning for Android malware classification demands a lot of resources, and it requires more time when more features are required for classification [14, 15]. Malware developers develop metamorphic and polymorphic approaches to evolve the malware and for evasion purposes. It is important to find minimal subsets of malware to identify malware effectively. Feature selection and optimization are challenging tasks [22]. Traditional adaptive neural networks are ineffective and suffer from low accuracy and precision. The minimal feature selection resolves the problem of considering many malware features for classification. Consequently, minimal feature selection for Android malware is applied first, and then classification is performed in the proposed approach. The proposed approach uses grey wolf

optimization for feature selection and optimization. The results demonstrate that ANFIS using grey wolf optimization has higher accuracy and precision.

GW optimizer is an evolutionary algorithm inspired by the social behavior of grey wolves. Grey wolves for hunting constitute groups and show discipline and cooperation [16]. Wolves are grouped into alpha, beta, delta, and omega. In the social community of wolves, they follow hierarchies like alpha (α) top in order, beta (β) second in line, delta (δ) third hierarchy in the community, and omega (ω) are the last group [16, 17]. Our proposed work has exploited the GWO to optimize Android malware features. Parallel exploration is capitalized to search for the best-suited subset of features for Android malware classification.

The remaining paper is organized as follows: Section 2 presents the Gray Wolf Optimizer (GWO) for ANFIS-AMAL and Android application features. Section 3 presents the Design and experimental setup of the ANFIS-AMAL, and Section 4 presents the experimental results and discussion of the study. The conclusion and future work are discussed in Section 5 of this paper.

2. Grey wolf optimizer for ANFIS-AMAL

The proposed GWO for ANFIS-AMAL is explained in this section. Firstly, a brief overview of the GWO is presented, and Android application features are discussed.

2.1. Grey wolf optimizer

GWO Algorithm solves optimization problems. It is inspired by nature, as wolves live and hunt in groups. They hunt in hierarchy by showing social behaviors of cooperation and living together. Healthy, active, and brave wolves are categorized as alpha in the group. Alpha wolves live the hunt and encircle the prey. The second category of wolves is beta wolves, which make the second circle around the prey. Similarly, the third circle of wolves is categorized as delta, and the fourth is omega. We have capitalized on the same hierarchy of grouping for selecting features for Android malware detection. The main features and effective features to categorize the Android malware classification are categorized as alpha(α). Similarly, relatively fewer groups of features are categorized as beta (β), delta (δ), and omega, respectively. This phenomenon helps to choose the most effective features. The presented work is an ensemble of this optimizer and the ANFIS approach.

Adaptive neuro-fuzzy inference uses the proposed technique's GWO Algorithm. In the currently proposed methodology context, groups of wolves are categorized into alpha, beta, delta, and omega. The proposed approach utilized GWO to explore the best features of Ransomware, Spyware, Botnet, and adware for Android. Exploration and exploitation of prey are different stages in the hunting process of grey wolves. $A > 1$ is for exploration of prey, i.e., searching the prey. $A < 1$ is for exploitation, i.e., attacking the prey.

The following are illustrations of the alpha (α), beta (β), delta (δ), and omega (ω) wolf represented candidates. The mathematical positioning of these groups of wolves is given in the equations below,

(1) $$\vec{L}_\alpha = \left|\vec{P}_1 \cdot \vec{X}_\alpha - \vec{X}\right|, \vec{L}_\beta = \left|\vec{P}_2 \cdot \vec{X}_\beta - \vec{X}\right|, \vec{L}_\delta = \left|\vec{P}_3 \cdot \vec{X}_\delta - \vec{X}\right|,$$

(2) $$\vec{X}_1 = \vec{X}_\alpha - \vec{\mathrm{A}}_1 \cdot \left(\vec{L}_\alpha\right), \vec{X}_2 = \vec{X}_\beta - \vec{\mathrm{A}}_2 \cdot \left(\vec{L}_\beta\right), \vec{X}_3 = \vec{X}_\delta - \vec{\mathrm{A}}_3 \cdot \left(\vec{L}_\delta\right),$$

(3) $$\vec{X}(i+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3},$$

where: $\vec{X}$ is the position vector of the wolf; i is the current iteration; $\vec{X}_\beta$ is the position vector of the beta wolf; $\vec{A}_1$, $\vec{A}_2$ and $\vec{A}_3$ are the coefficient matrix; $\vec{X}_\alpha$ denotes the position vector of the alpha wolf; p_i, i = 1, 2, 3, is the positioning of these groups of wolves is computed as illustrated in the above equations. The equations show mathematical computational formulas for positioning the wolves in different groups.

Other wolves in the population change positions according to these alpha, beta and delta groups.

In Android malware detection, the best candidate for Ransomware is alpha (α). Alpha candidate has the more prominent feature for the classification of malware. The proposed system explores features for Android malware in parallel. Multiple agents, i.e., candidate solutions, explore the best-suited attributes of malware. Once a better discriminatory feature is found, all other candidates converge to exploit it and find more features to distinguish the Android malware [18, 19].

GWO algorithm for ANFIS-AMAL uses entropy to compute the GWO algorithm's fitness function. As the equation below mentions, entropy is the fitness function computing formula. The first step is to compute feature information gain to calculate entropy. Entropy is important for classification to determine whether the purity of a feature, i.e., the feature, is important or not for the classification process, as shown in the next equation,

(4) $$\mathrm{E}\,(\mathrm{S}) = \sum[-p_i \log_2(p_i)],$$

where i = 1, …, n, and E represents entropy, and i is the total number of features, i.e., i represents iterations for the malware classes; e p_i denotes the number of elements in one class of malware.

Information gain calculation is based on equation,

(5) $$\mathrm{IG}\,(\mathrm{S}, \mathrm{A}) = \mathrm{E}(\mathrm{S}) - \sum\,[\mathrm{P}(\mathrm{S}/\mathrm{A}).\,\mathrm{E}(\mathrm{S}/\mathrm{A})].$$

IG – Information Gain;
E(S) – Entropy of Set S;
E(A) – Entropy of subset A;
P(S) – the Proportion of set S;
P(A) – the Proportion of set A.

The values of IG are used to split the set S, and a high value achieved for a feature of Android malware is selected for the optimal set of features.

2.2. Features of Android application used in ANFIS-MAL

The proposed work utilizes static and dynamic features and the Metadata of APK files. Permissions, intent, manifest information, and API calls are used along with the features in Table 1.

Table 1. Features of Android application used

Android application features	Description
WRITE_APN_SETTINGS	It allows the app to change APN settings. Useful Information like user name and password in the existing APN setting can be read
READ_PHONE_STATE	It permits read-only access to the state of the phone
MODIFY_PHONE_STATE	It allows changing the state of the phone
READ_SMS	This feature permits to read SMS messages
RESTART_PACKAGES	Allows to kill background process of other apps
RECIVE_BOOT_COMPLETED	Allows the app to receive the intent
WRITE_SMS	Allows to write SMS messages
MNT_UNMT_FILESYSTEMS	Allows the app to mount and un-mount file systems
INSTALLL_PACKAGES	Allows the app to install packages
BRDCAST_PCKGE_REMOVED	Permits the app to broadcast a notice that an app package is removed
INTERNET	Allows internet access
CHANGE_WIFII_STATE	Permits app to change Wi-Fi connectivity state
RECORDER_TASKS	Allows to record the task
SIG_STR	Allows to monitor phone signals
CALL_PHONE	Allows the app to start the call without dialer UI
KILL _BCKGRND_RESOURCES	Allows killing the background resources
BLUETOOTH	Allows Access to Bluetooth
CAMERA	Allows Access to camera
READ_CONTACTS	Allows the app to read the contacts
USER_CREDENTIALS	Allows access to the credentials of the user

Along with the features mentioned in Table 1 above, the proposed approach effectively uses some Ransomware features like crypto locker features and other type-specific features of botnets and spyware. Ransomware has features that goodware does not often have, like NtProtectVirtualMemory, NtResumeThread, and NtTerminateProcess. Consequently, these features lead the GWO to rank the malware as high risk because it is highly likely to be classified as Ransomware [20, 21]. The features are the most appropriate malware features used for Android malware classification. These features are used for the classification and identification process of Android malware.

The main set of features for classifying Android malware is selected and grouped based on the GWO algorithm and exploited for categorizing the risk level by the ANFIS algorithm. The proposed approach is an ensemble of two approaches: the first approach, GWO, is for exploring groups of Android malware features, while the second approach takes these groups of features and predicts the threat level of malware by exploiting ANFIS. Therefore, GWO is for exploration, and ANFIS is capitalized for exploitation in our proposed approach. An ensemble of these two approaches enables our presented work to classify Android malware more efficiently and effectively.

We evaluated several approaches for Android-based malware detection and classification. In our selection and evaluation criteria, we selected those approaches that utilize either machine learning or deep learning-based methodologies. We filtered both static and dynamic analysis-based approaches. Out of more than 200 approaches, we selected eight main approaches that have more relevance to our

selection criteria. We selected approaches that are more effective and efficient. Listed below are the main approaches in Table 2. The method, along with a brief description and main characteristics, are mentioned in the table below.

Table 2. Characteristics comparison of approaches in the literature

Authors	Method	Description of method	Main characteristics
Z h u, H., et al. [22]	DroidDet	Static analysis along with dynamic API analysis	The approach provides extraction of permissions and sensitive API monitoring system
Ç o b a n, O., S. A y s e [23]	Text categorization for manifest based	Adaptation for text analysis and categorization for Android malware utilizes static analysis	The technique provides Manifest-based text classification, it is efficient and utilizes low resources but provides high accuracy
S r e e j i t h, B. P., M. R. B a b u [14]	Ensemble-ANFIS Algorithm	The approach exploits PCA and PCC (Pearson Correlation Coefficient) beside ensemble ANFIS	The approach achieves comparatively high accuracy and efficiency, but it is complex and difficult to implement
A r i f, M., et al. [24]	Fuzzy AHP	Uses static features like permission-based analysis	Threat level assessment In four classes. This approach uses Multi-Criteria Decision-Making based (MCDM)
G a n d o t r a, E., D. B a n s a l, S. S o f a t [4]	Fuzzy logic paradigm for malware threat	Malware binaries analysis, potential threat capabilities analysis	Assessment of threat capability, malware binaries analysis using the fuzzy paradigm
L i n, Z., et al. [25]	Secure encryption-based malware detection	PP-NBC, API calls fragments analysis using Naive Bayes classifier	Anti-evasion technique, naïve Bayes classification, Encryption based malware analysis. It achieves high accuracy
R a z a, A., et al. [26]	Transfer learning	Dynamic analysis using transfer learning	A transfer learning-based approach to detect Android malware and classify different types of malware
Q a i s a r, Z. H., et al. [27]	Multi-agent based for mobile protection	Cloud protection for multi-agent based systems with encryption	Mobile cloud protection using an agent-based detection system. This approach is effective for detecting android malware in cloud environment

Ensemble-ANFIS and fuzzy AHP are more related to our criteria as both of these approaches use fuzzy logic. Ensemble ANFIS uses static analysis and exploits permission-based analysis for feature extraction. The fuzzy logic paradigm for malware threats uses binary analysis for malware analysis. Threat capability analysis is performed using the above-cited approach. All approaches were evaluated using our evaluation metrics and compared.

3. Design and working of ANFIS-AMAL

The ANFIS-AMAL is the ensemble of GWO and ANFIS. GWO is used for the selection of features for Android malware classification. Fig. 2 shows the overview of the proposed approach.

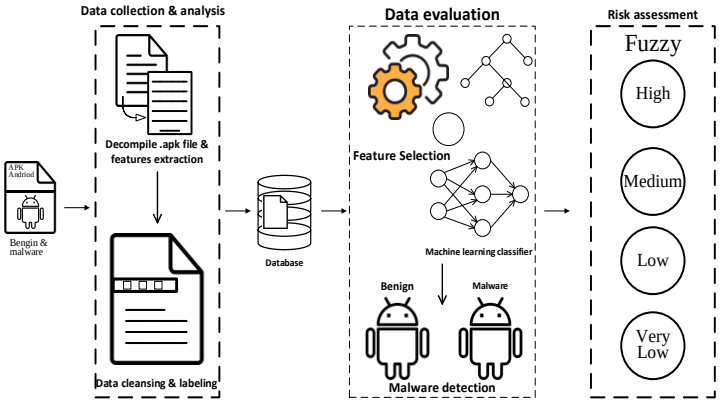

Fig. 2. Overview of ANFIS-AMAL

Risk assessment is categorized as very low, low, medium, and high. The proposed approach takes benign and malware Android apps. The minimal features are selected based on the Information Gain (IG) function. Information gain is based on entropy. The features are extracted and analyzed for evaluation. The extraction of features is performed by decompiling the APK. Then, data is analyzed using the rule inference system to classify input data [28]. Feature extraction and selection are important for learning.

3.1. Algorithm 1. GWO Algorithm for ANFIS-AMAL

The proposed work uses GWO to select features for Android malware classification explains the grey wolf optimizer. The algorithm takes features of Android malware as input. This algorithm effectively returns the optimal number of Android malware features for malware classification. Line 1 in Algorithm 1 represents the initialization of iterations *i* shows iterations, and *I* represents the total number of iterations. Line 2 indicates the randomly selected initialization population of wolves. The population of wolves is the Android malware feature in our work. Variables *a*, *A*, and *N* are tuning parameters adjusted to train the algorithm. $W\alpha$ represents the best candidate based on the information gain function. The best candidates in our work are the most suitable Android malware features. W_β denotes the second-best malware features, and W_δ represents the third-best features.

The loop block in Algorithm 1, starting from line 8 and ending at line 16, represents the iterative process of candidate selection and adjusting the parameters to select suitable candidates for malware classification. The fitness value is computed in each iteration to select the best candidate from the current iteration. In the end, the algorithm returns the most suitable malware features. Algorithm 1 GWO for ANFIS-AMAL is shown below.

Algorithm 1. GWO for ANFIS-AMAL

Input: Features of Android malware in vector format

Output: $\mathbf{W}_\alpha$, $\mathbf{W}_\beta$, and $\mathbf{W}_\delta$ groups for malware features

Step 1. Wp: initialize the population of grey wolves randomly in the search space;

Step 2. initialize the tuning parameters *a*, *A*, and *N*

Step 3. t_i: initialize the t=1, the iteration number

Step 4. IG: Compute the fitness of each grey wolf

Step 5. Select $\mathbf{W}_\alpha$, $\mathbf{W}_\beta$, and $\mathbf{W}_\delta$ from $\mathbf{W}_p$ (p =1, 2,..., N);

Step 6. Tuning parameters *a*, *A*, and *N*;

Step 7. calculate the objective function Info Gain (IG) value of each grey wolf;

Step 8. $\mathbf{W}_\alpha$ = the best candidate based on Info Gain (IG);

Step 9. $\mathbf{W}_\beta$ = the second-best candidate;

Step 10. $\mathbf{W}_\delta$ = the third best candidate;

Step 11. **while** (t_i < I)

Step 12. **for** each candidate wolf

$$X(t_i + 1) = \frac{X_1 + X_2 + X_3}{3}$$

Step 13. update the position of the candidate wolves based on fitness;

Step 14. **end for;**

Step 16. compute the Fitness values of all grey wolves based on IG;

Step 17. update $\mathbf{W}_\alpha$, $\mathbf{W}_\beta$, and $\mathbf{W}_\delta$;

Step 18. $t_i = t_i + 1$;

Step 19. **end while;**

The algorithm is used for the selection of appropriate features. As explained previously, the algorithm aims to extract the minimal set of features for Android malware, effective enough to classify and distinguish malware. The selection of features is based on fitness features [29]. High IG (Information Gain) helped in the selection of features as α (alpha), and similarly, low IG will classify features as β (beta) or δ (delta).

The first three lines of Algorithm 1 represent the initialization process of the algorithm. The selection of alpha, beta, and delta groups for wolves from the population of wolves is explained in the remaining statements of the algorithm. Line 12 of the Algorithm 1 explains the computation of the positioning of the wolves. The positions are updated based on the formula concerning time. The output of the algorithm is groups of wolves based on fitness functions.

The enriched and diversified data sets are used for training and validation, as mentioned in Table 3. The enriched and diversified datasets are effective for the learning and classifying of Android malware, as mentioned in datasets utilized for training and validation.

Table 3. Datasets utilized for training and validation

Data set	Malicious samples	Normal samples	Total
Malgenome [30]	1258	37,627	38,885
Drebin [31]	5555	37,627	43,182
MalDozer [32]	20,089	37,627	57,716

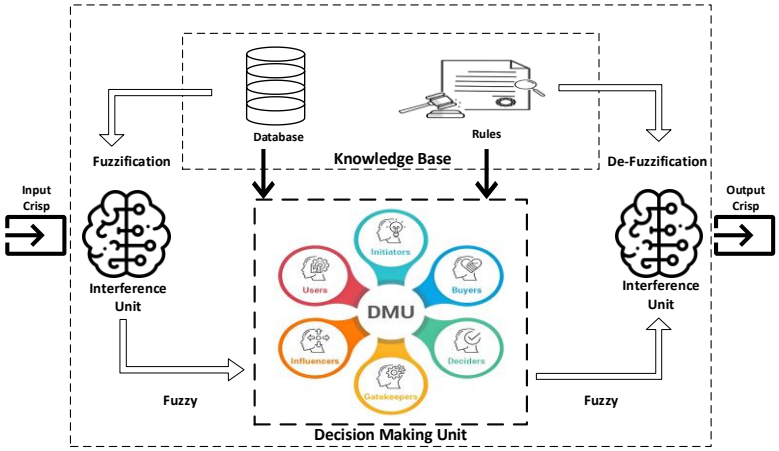

Fig. 3. General steps for proposed ANFIS-AMAL

Three main different kinds of Android malware are listed in Table 3. Malgenome, Drebin and MaldDozer. Datasets contain benign and malware samples. These datasets are for training and validation. The steps of ANFIS-AMAL for Android risk assessment or threat ranking are mentioned in Fig. 3.

In the proposed work, hybrid malware analysis is used. As mentioned above, the proposed technique uses benign and malware data sets for training and validation. Both static and dynamic malware features and some metadata information of APK files are used.

3.2. Layers of proposed ANFIS-AMAL

The five layers of the adaptive neuro-fuzzy inference system are shown in Fig. 4 below. The computational formula for each layer is also represented in the equations given below. Layers are adjustable and number of neurons in each layers are also based on heuristic approach. The proposed ANFIS takes the output of the GWO and performs analysis to determine the impact of malware. Threat assessment and risk assessment is determined using the ANFIS approach.

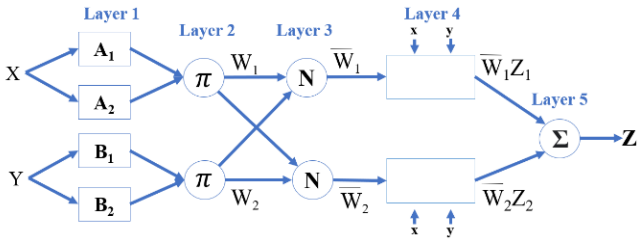

Fig. 4. Structure of layers of ANFIS-AMAL

The computational model applied at each layer of ANFIS-AMAL is proposed here. The mathematical model for layer one is illustrated in the next equation,

$$O_{mn}^{(1)} = \mu_{A_{mn}}(m). \quad (6)$$

A computational model for layer two is illustrated in the next equation:

(7) $$O_k^{(2)} = w_k = \prod_{i=1}^{7\ \text{inputs}} \mu_{A_{ij}}(I_i).$$

The mathematical model for layer three is depicted in the equation

(8) $$O_k^{(3)} = \overline{w_k} = \frac{w_k}{\sum_{k=1}^{\text{NumRules}} w_k}.$$

The mathematical model for layer four is shown in the equation

(9) $$O_k^{(4)} = \overline{w_k} f_k = \overline{w_k}\left(\sum_{i=1}^{7} p_{ik} I_i + q_k\right).$$

The mathematical model for layer five is given in the equation

(10) $$O^{(5)} = \sum_{k=1}^{\text{NumRules}} \overline{w_k} f_k = \frac{\sum_{k=1}^{\text{NumRules}} w_k f_k}{\sum_{k=1}^{\text{NumRules}} w_k}.$$

As mentioned in the equation below, the hybrid-learning algorithm is depicted mathematically in the next equation. The calculated value is forwarded to output nodes to the 4th layer of the ANFIS-AMAL model [33, 34]. ANFIS-AMAL exploits cited a hybrid-learning algorithm for learning the features of threat assessment of Android malware:

(11) $$L = \sum_{x=1}^{N} \overline{W_x} \left(\sum_{j=1}^{\text{Input}} M I_j + N_x\right).$$

The least-square method is used for the parameters. Gradient descent is for adjustment of parameters, and back-propagation of error is applied [35].

3.2.1. De-fuzzification layer

The De-fuzzification layer in the proposed approach has all nodes as adaptive layer nodes. The p represents the nodes in the layer. The output layer is followed by a normalized product with an overall accumulative effect and a first-order polynomial defined in the equation [36][37]. Layer output can be illustrated in the given equation

(12) $$\text{Out}_p^{(4)} = \overline{w_R} f_R = \overline{w_R}\left(\sum_{i=1}^{\text{rules}} M_{iR} I_i + N_R\right),$$

where, $R = 1, 2, \ldots, 1247$ onwards rules ,

$\overline{w_R}$ = Activation value normalized from the previous layer,

M_{iR} and N_R are output parameters of the system and tuned parameters of the algorithm, respectively.

The following are rules Rule 1 and Rule 2, which show the incident and consequent part and inference of rules. These rules are used for inference systems in the proposed model.

Rule 1: IF x is A_1 AND y is B_1, THEN

$$f_1 = p_1 x + q_1 y + r_1$$

Rule 2: IF x is A_2 AND y is B_2, THEN

$$f_2 = p_2 x + q_2 y + r_2$$

3.2.2. Android malware threat scaling

Android malware threat scaling is adopted in the proposed approach. The ranking system is based on ten (10) points. The ten-point score is normalized from 0 up to 1. Point scoring is used for malware ranking. The malware with a high score is ranked in the high-threat category. In Fig. 5, risk level and risk value are illustrated. The four main categories are considered based on risk value: very low, low, medium, and high risk, as shown in the diagram with different colors based on risk value [38, 39].

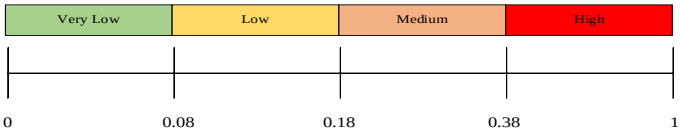

Fig. 5. Threat scaling for Android malware

3.3. Tuning details of ANFIS-AMAL

Fuzzy inference systems are applied to problems with vagueness and blurred domain boundaries. The threat assessment problem has a similar vagueness issue, so it is a candidate for FIS [40, 41]. The proposed framework is implemented in MATLAB2022b. The implemented system used Intel Core i7-6500U with a CPU of 2.60 GHz and 8 GB RAM. Fig. 6 illustrates the implementation of membership functions for input and out of the proposed system. As fuzzy set membership differs from crisp sets, different modeling approaches are used for implementation [40, 42]. The proposed approach uses a triangular membership function for the output variable and various appropriate membership functions for input [43, 44]. The output of the proposed ANFIS-AMAL uses a de-fuzzified using de-fuzzification process [45, 46]. A rule-based inference system is used to infer the threat category of the Android malware. Input and output are plotted using membership functions for inputs and outputs in the proposed model. The input is fuzzified, while the output is de-fuzzified to extract the results. Each layer in the ANFIS model has a defined role.

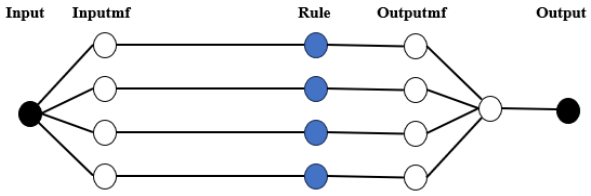

Fig. 6. ANFIS-AMAL generated model

The tuning parameters and architecture of ANFIS-AMAL are mentioned in Table 4. The table shows the parameters for tuning and structure of the proposed ANFIS-AMAL. Model type, input and training, and the testing mechanism are listed in the same table.

Table 4. ANFIS-AMAL model description

No	Custom ANFIS	Details
1	Number of Training data	92,000 records
2	Number of Testing data	21,000 records
3	Generate FIS	Grid partition
4	Inputs	GWO-based features, static and dynamic APK features
5	Membership function type	Triangular-shaped membership function
6	Learning algorithm	Hybrid learning algorithm
7	Number of epochs	Ten (10) then increased to 40 and 100 epochs
8	Sugeno type-system	First-order
9	Hybrid learning algorithm	Least-square method and Back-propagation gradient descent method

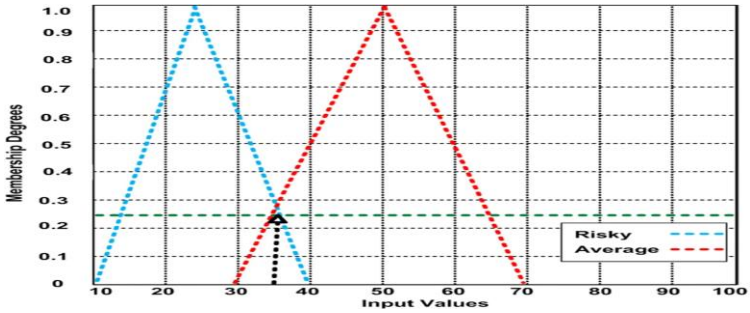

Fig. 7. Membership functions of ANFIS-AMAL

The modeling of the membership function in MATLAB is depicted in Fig. 7. Appropriate membership functions are used in modeling the adaptive neuro-fuzzy inference system. The input value is plotted quantitatively on the *X*-axis, and the membership degrees are shown along the *Y*-axis. The risk level is represented in different colors to make it distinguishable. Unlike crisp sets here, overlapping can be observed for the fuzzy sets. Different membership functions are used for different inputs, depending on the problem. The degree of membership varies from 0 up to 1. This variation in the degree of the membership, which is quantitative, can be translated into the equivalent qualitative classes. Then, these qualitative classes can be used to define the malware threat level in our work. Different membership functions are used for different inputs, depending on the problem. The degree of membership varies from 0 up to 1. This variation in the degree of the membership, which is quantitative, can be translated into the equivalent qualitative classes. Then, these qualitative classes can be used to define the malware threat level in our work.

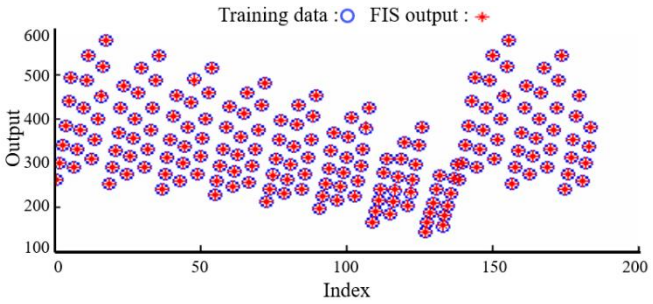

Fig. 8. Loaded Training dataset into ANFIS

ANFIS-AMAL is trained for classifying the threat of malware. Main Three categories, i.e., low threat, moderate threat, and high threat for Android malware, are applied. Fig. 8 shows the training of the proposed system. The oval structures in blue represent training data, and asterisks in red show FIS output. Mapping of training data and FIS output shows the training process is reliable. Training data and output data are plotted in this figure to show the training mechanism. It shows the training is effective with low training error for different indexes.

4. Results and discussion

ANFIS-AMAL uses a ten-point Likert scale for the point-scoring system of Android malware threats. The scale is normalized, and values are confined to 0 up to 1. Fig. 9 shows ANFIS-AMAL errors at the start and middle at 40 epochs, and Fig. 10 shows ANFIS-AMAL errors at 100 epochs. Training required 105 epochs. The training process at the start and 40 epochs is shown in the figures. The proposed ANFIS-AMAL has taken less time and fewer epochs than traditional ANFIS because of the ensemble of GW. The GW optimizer has enabled the proposed work to learn fast and train in fewer epochs with better results. It can be observed that the error rate is high at the start, and then it tends to decrease. In our proposed work, GWO has provided the optimal number of Android malware features for classification. Therefore, fewer features and effective features in the learning process result in low training error by consuming fewer epochs for training.

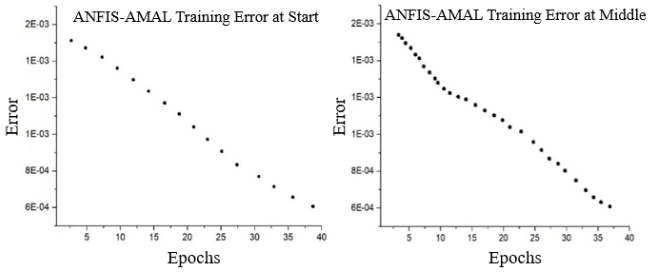

Fig. 9. ANFIS-AMAL training error at start and middle

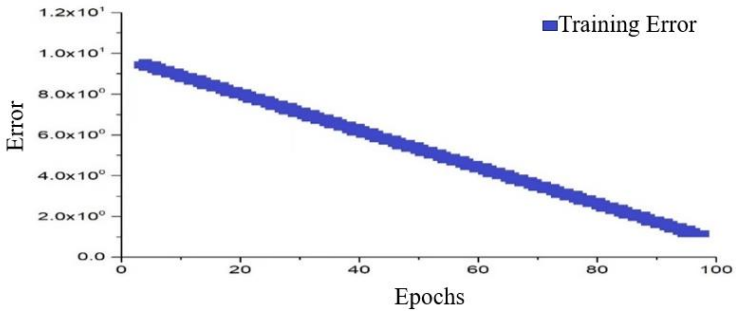

Fig. 10. ANFIS-AMAL training error at end

The testing process of ANFIS-AMAL is shown in Fig. 11. Testing data is plotted in blue dots; however, the FIS output is represented in asterisks. The plotting shows the validation process for our proposed model. At the start, the training error is greater; in the middle, it decreases, and at the end of 100 epochs, it becomes minimal. A relatively minimal number of epochs are consumed to get the required modest training error rate and precise results. This is due to the reduced number of features selected through the GWO algorithm. Despite GWO's processing complexity, the proposed work gains high accuracy and efficiency. So, there is a trade-off between the processing cost of GWO and the better performance of

ANFIS-AMAL. For training and testing, different data is used so that effective training and testing can be performed without bias.

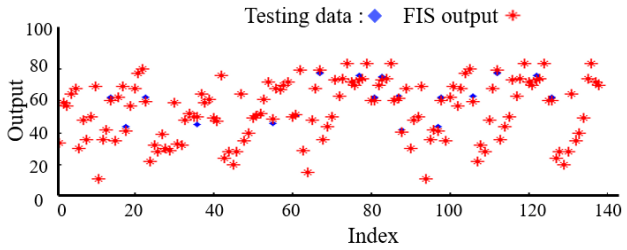

Fig. 11. ANFIS-AMAL Model testing with testing data

Experimental results show that our proposed ANFIS-AMAL is efficient and effective in assessing the threat of Android malware. The demonstrated model has attained an accuracy of 94%, a precision of 93%, and a sensitivity of approximately 95%. The results reveal that an excellent F-measure score of 95% is obtained. These results show less false-negative rate and confirm that the overall model is effective for Android malware assessment employed. The results show that the system is reliable and comparable.

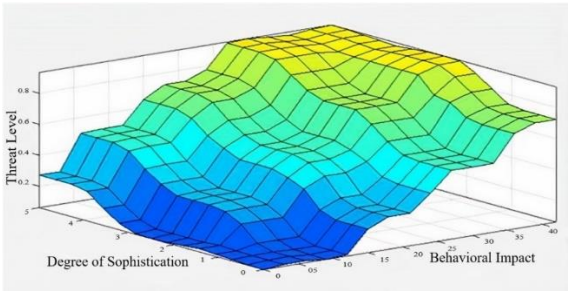

Fig. 12. Threat level of the malware plotted as output relation with input variables

Fig. 12 shows the threat level of the malware plotted concerning input variables. The threat level is shown quantitatively concerning malware's degree of sophistication and behavioral impact. This figure shows the relationship between input variables (behavioral impact and degree of sophistication) and an output variable (malware threat level). The threat level ranges from 0 up to 1. The input variables are fuzzified using different membership functions, as discussed earlier. Table 5 below shows the error rate at various epochs in the RMSE proposed model's training, validation, and testing. The training error of 0.0009, validation error of 0.11, and testing error of 0.0023 are acceptable for ANFIS for the currently proposed system and are approximately similar in the resembling systems [47, 48].

Table 5. Error rate at various epochs

Number of training epochs	Training error rate
10	0.0010370
40	0.00099596
100	0.00090881

Error is assessed at the 10-th, 40-th and 100-th epoch, and the training error rate in RMSE is assessed in the above table.

4.1. Evaluation metrics

ANFIS-AMAL is assessed on four metrics: accuracy, specificity, precision, and F-score. The equations below mention the computing method for calculating specificity, precision, and accuracy. The effectiveness and efficacy of the proposed system are assessed by exploiting these metrics. Measurement of the threat is based on points using a ten-point Likert scale. The degree of threat is then translated into qualitative measures using the de-fuzzification mechanism to declare malware as a high threat, moderate threat, or low threat based on numeric values.

The evaluation metrics have been applied to the proposed work. Accuracy is approaching 95.46%, while sensitivity is above 94.23%. It has a precision of 94.41% and a commendable F-score of 95.36%. Literature for modeling the adaptive neuro-fuzzy system shows that these results are dependable [3, 15]. Metrics of evaluation are defined in the following subsections.

Specificity represents the percentage of benign apps correctly classified by the system. Specificity is defined in equation

$$\text{Specificity} = \frac{\text{True Negative}}{\text{True Negative + False Positive}} \times 100\ \%. \tag{13}$$

Evaluation metrics specificity mentioned in that equation and Accuracy, precision, recall, F-measure, FPR, and TPR are used to assess the proposed approach.

4.2. Comparison of results

The comparison of the proposed ANFIS-AMAL with other state-of-the-art approaches is illustrated below. The proposed approach, ANFIS-AMAL, is compared with Fuzzy-AHP and ensemble-ANFIS [29, 28]. The risk-based fuzzy analytical hierarchy process (AHP) is compared with the proposed ANFIS-AMAL approach. Fig. 13 shows that Fuzzy-AHP has a specificity of 87.12%, a precision of 88.24%, an accuracy of 89.10%, and an F-score of 88.42%. Similarly, the previous approach, ensemble-ANFIS, has 87.53% accuracy, 86.52% precision, 84.14% specificity, and 88.16% F-score. The proposed approach has outperformed in these evaluation metrics. Proposed ANFIS-AMAL has a better accuracy of 95.46%, a specificity of 94.23%, a precision of 94.41%, and an F-score of 95.36%. We applied an ablation study to compare the proposed approach results with excluding GWO. The results of the ablation study show that our approach is effective with the inclusion of GWO and has better results than the approach without GWO.

The proposed work is compared with the Fuzzy-AHP and ensemble-ANFIS. Inter-class and intra-class threat assessment using ANFIS-AMAL is performed. Similar results are achieved for inter-class and intra-class threat assessment for Android malware. Evaluation metrics-based results are better for inter-class malware threat analysis, but for intra-class, the results are relatively low. The intra-class feature distinguishing is challenging as within the same class, ranking features is difficult. However, inter-class results are commendable as it is easy to distinguish different classes of malware based on features. Inter-class threat assessment for Android malware is better using the proposed work. Ransomware, botnets, spyware,

and adware are detected and analyzed, and threat ranking is performed and categorized in the proposed threat model.

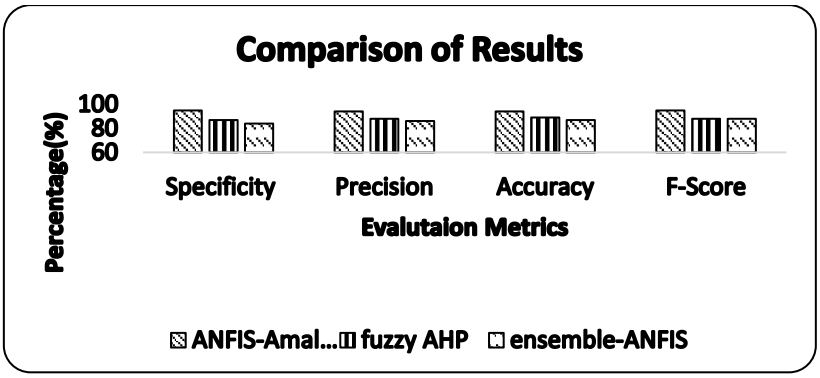

Fig. 13. Comparison of proposed ANFIS-AMAL with other approaches

As listed in Table 3 Malgenome, Drebin, and Maldozer all three benchmark datasets are skewed because benign apps are more than malware apps. To remove skewness we selected an equal number of malware and benign apps so that there should be fairness in experiments. In our experimental setup, we have employed an ablation study. Purpose of ablation study is to help in evaluation of ensemble and show effect of ensemble in results. Results show that evaluation metrics for with ensemble show better results as compare to the results without ensemble.

Table 6. Ablation Study for Performance with and without Ensemble Learning

Model	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)	AUC	Time (ms)
ANFIS-without Ensemble	97.85	97.95	97.42	97.53	0.98	5.04
ANFIS-with Ensemble	98.07	99.00	98.00	98.32	0.99	12.19

Illustrated in Table 6 the impact of ensemble learning improves results for Accuracy, precision, Recall, F-measure, and AUC. However, it takes more time for ANFIS with the ensemble as compared to ANFIS without an ensemble but better results overcome this deficiency. Results were evaluated using more than 10 evaluation metrics like accuracy, precision, recall, F-measure, time, True Positive Rate (TPR), False Positive Rate (FPR), Area Under the Curve Receiver Operating Characteristic (AUC-ROC), Area Under the Curve Precision-Recall (AUC-PR), Matthews Correlation Coefficient (MCC), detection rate, and AUC. However, for brevity, we have illustrated results for some evaluation metrics that are more worth mentioning; other evaluation metrics were either utilized in computing these illustrated metrics or had a nominal or negligible impact on the results.

4.3. Analysis of proposed work

The proposed work has some limitations as it is based on the scaling of threats and results that are sensitive to the accuracy of the points-scoring process of the ANFIS-

AMAL. The scoring point is mapped on the scaling system. If the point scoring system's accuracy is compromised, it will affect the results of the proposed system. The second limitation of the proposed work is that the GWO algorithm has its complexity of implementation and resource consumption [35, 49]. However, overall, ANFIS-Mal is efficient and more accurate because of the ensemble of the GW optimization algorithm. However, extra processing and layers of GWO make the system complex, so it's a trade-off between complexity and efficiency.

5. Conclusions and future work

The ensemble of GWO and ANFIS is applied in Android malware threat prioritization in the proposed work. The adaptive neuro-fuzzy inference is effective in qualitative analysis, and this advantage of ANFIS is exploited in the proposed work. The ensemble of Grey Wolf Optimizer is used in the proposed work to enhance the efficiency and effectiveness of the ANFIS-AMAL in malware threat assessment. Ensemble learning and ensemble approaches are more effective in learning and classification problems. The proposed approach uses GWO to enhance learning abilities and optimize the feature selection process to select a minimal number of features for Android malware classification. The fitness function in GWO is essential in selecting the most appropriate and ultimately minimal set of Android malware features. ANFIS uses a minimal set of features to classify the threat. The threat is categorized into three main categories: high, medium, and low, with a variant of very low. Ransomware, botnets, spyware, and adware are categorized in the proposed work. Proposed ANFIS-AMAL has attained a better accuracy of 95%, specificity of 94%, Precision of 94%, and F-score of 95%.

In the proposed work, the priority for Ransomware is high, with botnets medium and spyware low. These priorities are used to evaluate the proposed work, but Android malware priorities can be changed and adjusted. Results are evaluated based on evaluation metrics. The results are compared with the previous approaches in the literature. Results show that the proposed work is more effective and has high efficacy. Threats of Android malware are assessed with acceptable results with fast learning and optimized feature selection. Fewer epochs are consumed in training through the ensemble of GWO for learning, and fewer resources are consumed in ANFIS. Consequently, efficient learning and high accuracy, specificity, precision, and F-score are salient contributions to the proposed ANFIS with the ensemble of GWO.

The proposed work can be extended because Android malware can have overlapping features and dual behavior. Ransomware can belong to Ransomware and botnet as well. This duality can mislead the proposed work as Ransomware has high priority and botnet has low, so in this case, the inter-class threat should be implemented in detail. In this way, inter-class Android malware threat prioritization can be improved.

References

1. Nadler, A., R. Bitton, O. Brodt, A. Shabtai. On the Vulnerability of Anti-Malware Solutions to DNS Attacks. – Computers & Security, Vol. **116**, 2021, pp. 1-16.
2. Shanmuganathan, M., Majdi Al-qdah, M. Mohmmed, C. Narmatha, R. Varatharajan. Intrusion Detection in Networks Using Crow Search Optimization Algorithm with Adaptive Neuro-Fuzzy Inference System. – Microprocessors and Microsystems, Vol. **79**, 2020, No 8, pp. 1-7.
3. Liu, J., W. Yinchai, T. C. Siong, X. Li, L. Zhao, F. Wei. A Hybrid Interpretable Deep Structure Based on Adaptive Neuro-Fuzzy Inference System, Decision Tree, and k-Means for Intrusion Detection. – Scientific Reports, Vol. **12**, 2022, No 1, 20770.
4. Gandotra, E., D. Bansal, S. Sofat. Malware Threat Assessment Using Fuzzy Logic Paradigm. – Cybernetics and Systems, Vol. **48**, 2017, No 1, pp. 29-48.
5. Ayeni, B. K., J. B. Sahalu, K. R. Adeyanju. Detecting Cross-Site Scripting in Web Applications Using Fuzzy Inference System. – Journal of Computer Networks and Communications, Vol. **2018**, 2018, pp. 1-11.
6. Abdulla, S., A. Altaher. Intelligent Approach for Android Malware Detection. – KSII Transactions on Internet and Information Systems, Vol. **9**, 2015, No 8, pp. 2964-2983.
7. Qaisar, Z. H., R. Li. Multimodal Information Fusion for Android Malware Detection Using Lazy Learning. – Multimedia Tools and Applications, 2022, pp. 1-15.
8. Altaher, A., O. Mohammed. Intelligent Hybrid Approach for Android Malware Detection Based on Permissions and API Calls. – International Journal of Advanced Computer Science and Applications, Vol. **8**, 2017, No 6, pp. 60-67.
9. Santosh Jhansi, K., S. Chakravarty, P. Ravi Kiran Varma. A Two-Tier Fuzzy Meta-Heuristic Hybrid Optimization for Dynamic Android Malware Detection. – SN Computer Science, Vol. **4**, 2022, No 2, pp. 117.
10. Xu, Y., C. Wu, K. Zheng, X. Wang, X. Niu, T. Lu. Computing Adaptive Feature Weights with PSO to Improve Android Malware Detection. – Security and Communication Networks, Vol. **2017**, 2017, pp. 1-14.
11. Sun, H., G. Xu, Z. Wu, R. Quan. Android Malware Detection Based on Feature Selection and Weight Measurement. – Intelligent Automation and Soft Computing, Vol. **33**, 2022, No 1, pp. 585-600.
12. Lakovic, V. Crisis. Management of Android Botnet Detection Using Adaptive Neuro-Fuzzy Inference System. – Annals of Data Science, Vol. **7**, 2020, No 2, pp. 347-355.
13. Liu, J., W. Yinchai, T. C. Siong, X. Li, L. Zhao, F. Wei. On the Combination of Adaptive Neuro-Fuzzy Inference System and Deep Residual Network for Improving Detection Rates on Intrusion Detection. – PLOS ONE, Vol. **17**, 2022, No 12, pp. 1-21.
14. Sreejith Vignesh, B. P., M. Rajesh Babu. Classifying the Malware Application in the Android-Based Smartphones Using Ensemble-ANFIS Algorithm. – International Journal of Networking and Virtual Organisations, Vol. **19**, 2018, No 2-4, pp. 257-269.
15. Atacak, İ. An Ensemble Approach Based on Fuzzy Logic Using Machine Learning Classifiers for Android Malware Detection. – Applied Sciences, Vol. **13**, 2023, No 3.
16. Rezaei, H., O. Bozorg-Haddad, X. Chu. Grey Wolf Optimization (GWO) Algorithm. – In: Studies in Computational Intelligence. Vol. **720**. Springer, 2018, pp. 81-91.
17. Taher, F., O. Al Fandi, M. Al-kfairy, H. Al Hamadi, S. Alrabaee. DroidDetectMW: A Hybrid Intelligent Model for Android Malware Detection. – Applied Sciences, Vol. **13**, No 13, 2023.
18. Huang, H., H. Deng, Y. Sheng, X. Ye. Accelerating Convolutional Neural Network-Based Malware Traffic Detection through Ant Colony Clustering. – Journal of Intelligent and Fuzzy Systems, Vol. **37**, 2019, No 1, pp. 409-423.
19. Dhalaria, M., E. Gandotra. Android Malware Risk Evaluation Using Fuzzy Logic. – In: Proc. of 7th International Conference on Parallel, Distributed and Grid Computing (PDGC'22), 2022, pp. 341-345.

20. X i a o, G., J. L i, Y. C h e n, K. L i. MalFCS: An Effective Malware Classification Framework with Automated Feature Extraction Based on Deep Convolutional Neural Networks. – Journal of Parallel and Distributed Computing, Vol. **141**, 2020, pp. 49-58.
21. S h a r m a, A., N. K a p o o r. Approach for Predicting Mobile Malware. – In: Proc. of 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N'22), 2022, pp. 1614-1618.
22. Z h u, H. J., Z. H. Y o u, Z. X. Z h u, W. L. S h i, X. C h e n, L. C h e n g. DroidDet: Effective and Robust Detection of Android Malware Using Static Analysis Along with Rotation Forest Model. – Neurocomputing, Vol. **272**, 2018, pp. 638-646.
23. Ç o b a n, Ö., S. A. Ö z e l. Adapting Text Categorization for Manifest Based Android Malware Detection. – Computer Science, Vol. **20**, 2019, No 3, pp. 483-405.
24. M o h a m a d A r i f, J., M. F. A b R a z a k, S. R. T u a n M a t, S. A w a n g, N. S. N. I s m a i l, A. F i r d a u s. Android Mobile Malware Detection Using Fuzzy AHP. – Journal of Information Security and Applications, Vol. **61**, 2021, No 1, pp. 1-11.
25. L i n, Z., F. X i a o, Y. S u n, Y. M a, C. C. X i n g, J. H u a n g. A Secure Encryption-Based Malware Detection System. – KSII Transactions on Internet and Information Systems, Vol. **12**, 2018, No 4, pp. 1799-1818.
26. R a z a, A., M. F a h e e m, M. W. A s h r a f, M. N. C h a u d h r y. TL-GNN : Android Malware Detection Using Transfer Learning. – Applied AI Letters, Vol. **1**, 2024, pp. 1-16.
27. Q a i s a r, Z. H., S. H. A l m o t i r i, M. A. A l G h a m d i, A. A. N a g r a, G. A l i. A Scalable and Efficient Multi-Agent Architecture for Malware Protection in Data Sharing over Mobile Cloud. – IEEE Access, Vol. **9** , 2021, pp. 76248-76259.
28. J e r b i, M., Z. C h e l l y D a g d i a, S. B e c h i k h, L. B e n S a i d. Android Malware Detection as a Bi-Level Problem. – Computers & Security, Vol. **121**, 2022, 102825.
29. A d e b a y o, O. S., N. A. A z i z. Improved Malware Detection Model with Apriori Association Rule and Particle Swarm Optimization. – Security and Communication Networks, Vol. **2019**, 2019, pp. 1-14.
30. Z h o u, Y., X. J i a n g. Dissecting Android Malware: Characterization and Evolution. – In: Proc. of IEEE Symposium on Security and Privacy, 2012, No 4, pp. 95-109.
31. A r p, D., M. S p r e i t z e n b a r t h, M. H ü b n e r, H. G a s c o n, K. R i e c k. Drebin: Effective and Explainable Detection of Android Malware in Your Pocket. – In: Proc. of NDSS Symposium, 2014, pp. 23-26.
32. K a r b a b, E. B., M. D e b b a b i, A. D e r h a b, D. M o u h e b. MalDozer: Automatic Framework for Android Malware Detection Using Deep Learning. – Digital Investigation, Vol. **24**, 2018, pp. S48-S59.
33. R e j e e s h, M. R. Interest Point Based Face Recognition Using Adaptive Neuro-Fuzzy Inference System. – Multimedia Tools and Applications, Vol. **78,** 2019, No 16, pp. 22691-22710.
34. V u, N. T. T., N. P. T r a n, N. H. N g u y e n. Adaptive Neuro-Fuzzy Inference System Based Path Planning for Excavator's Arm. – Journal of Robotics, Vol. **2018**, 2018, pp. 1-7.
35. F a n g, L., S. D i n g, J. H. P a r k, L. M a. Adaptive Fuzzy Control for Stochastic High-Order Nonlinear Systems with Output Constraints. – IEEE Transactions on Fuzzy Systems, Vol. **29**, 2021, No 9, pp. 2635-2646.
36. C ó z a r, J., A. F e r n á n d e z, F. H e r r e r a, J. A. G á m e z. A Metahierarchical Rule Decision System to Design Robust Fuzzy Classifiers Based on Data Complexity. – IEEE Transactions on Fuzzy Systems, Vol. **27** , 2019, No 4, pp. 701-715.
37. M u z a f f a r, A., H. R a g a b H a s s e n, M. A. L o n e s, H. Z a n t o u t. An In-Depth Review of Machine Learning Based Android Malware Detection. – Computers & Security, Vol. **121**, 2022, 102833.
38. W u, Y., M. L i, Q. Z e n g, et al. DroidRL: Feature Selection for Android Malware Detection with Reinforcement Learning. – Computers & Security, Vol. **128**, 2023, 103126.
39. A l-A n d o l i, M. N., S. C. T a n, K. S. S i m, C. P. L i m, P. Y. G o h. Parallel Deep Learning with a Hybrid BP-PSO Framework for Feature Extraction and Malware Classification. – Applied Soft Computing, Vol. **131**, 2022, 109756.
40. G a s c o n, H., F. Y a m a g u c h i, D. A r p, K. R i e c k. Structural Detection of Android Malware Using Embedded Call Graphs. – In: Proc. of ACM Conference on Computer and Communications Security, 2013, pp. 45-54.

41. A r s l a n, R. S., M. T a s y u r e k. AMD-CNN: Android Malware Detection via Feature Graph and Convolutional Neural Networks. – Concurrency and Computation: Practice and Experience, Vol. **34**, 2022, No 23, e7180.
42. A n d e r s o n, H. S., P. R o t h. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. – ArXiv, 2018, pp. 1-8.
http://arxiv.org/abs/1804.04637
43. P a t r o, S. G. K., B. K. M i s h r a, S. K. P a n d a, R. K u m a r, H. V. L o n g, T. M. T u a n. Knowledge-Based Preference Learning Model for Recommender System Using Adaptive Neuro-Fuzzy Inference System. – Journal of Intelligent & Fuzzy Systems, Vol. **39**, 2020, No 3, pp. 4651-4665.
44. D h a b a l, G., G. G u p t a.Towards Design of a Novel Android Malware Detection Framework Using Hybrid Deep Learning Techniques. – In: Soft Computing for Security Applications. Singapore, Springer Nature, 2023, pp. 181-193.
45. K a r a h o c a, A., D. K a r a h o c a. GSM Churn Management by Using Fuzzy c-Means Clustering and Adaptive Neuro-Fuzzy Inference System. – Expert Systems with Applications, Vol. **38**, 2011, No 3, pp. 1814-1822.
46. G e z e r, A., G. W a r n e r, C. W i l s o n, P. S h r e s t h a. A Flow-Based Approach for Trickbot Banking Trojan Detection. – Computers and Security, Vol. **84**, 2019, pp. 179-192.
47. W a n g, Y.-M., T. M. S. E l h a g. An Adaptive Neuro-Fuzzy Inference System for Bridge Risk Assessment. – Expert Systems with Applications, Vol. **34**, 2008, No 4, pp. 3099-3106.
48. A l a m r o, H., W. M t o u a a, S. A l j a m e e l, A. S. S a l a m a, M. A. H a m z a, A. Y. O t h m a n. Automated Android Malware Detection Using Optimal Ensemble Learning Approach for Cybersecurity. – IEEE Access, Vol. **11**, 2023, pp. 72509-72517.
49. M e l a n d, P. H., Y. F. F. B a y o u m y, G. S i n d r e. The Ransomware-as-a-Service Economy Within the Darknet. – Computers and Security, Vol. **92**, 2020, No 7034, pp. 1-9.

Received: 13.12.2023; Second Version: 27.04.2024; Third Version: 24.07.2024; Accepted: 08.08.2024