Open Access Article

WRAPPER-BASED METAHEURISTIC OPTIMIZATION ALGORITHMS FOR ANDROID MALWARE DETECTION: A CORRELATIVE ANALYSIS OF FIREFLY, BAT & WHALE OPTIMIZATION

Santosh Jhansi K
Centurion University of Technology and Management, Odisha, India

Sujatha Chakravarthy
Centurion University of Technology and Management, Bhuvaneswar, India

Ravi Kiran Varma P
Maharaj Vijayaram Gajapathi Raj College of Engineering, Vizianagaram, India

Abstract: Recently, Android malware threats have been increasing at a rapid pace along with their usage and popularity. Different studies have elucidated the importance of analyzing the Android permissions pattern for the effective detection of Android malware. Optimization of android malware detection with high-dimensional permissions data is a bottleneck, which is a burning issue. In this study, nature-inspired wrapper-based metaheuristic algorithms such as firefly, bat, and whale optimization algorithms are investigated for the analysis of different Android permission patterns suitable for malware detection. Different machine learning classification algorithms, such as Linear Regression (LR), Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Decision Tree (DT), Random Forest (RF), Gradient Boosting (GB), and Extreme Learning Machine (ELM) are employed for evaluation. The Wrapper-Based Feature Selection using the Firefly Algorithm (WFSFA) outperformed the other algorithms in terms of feature reduction and recorded an improved classification accuracy of 95.28% when experimented with high-dimensional CICInvesAndMal2019 feature dataset with 4115 features.

Keywords: Android Security, Malware Analysis, Static Analysis, Feature Selection, Android Malware Classification, Firefly Algorithm (FA), Bat Algorithm (BA), Whale Algorithm (WA).

摘要：**最近**，Android 恶意软件威胁随着它们的使用和流行而迅速增加。不同的研究阐明了分析 Android 权限模式对于有效检测 Android 恶意软件的重要性。用高维权限数据优化android恶意软件检测是一个瓶颈，这是一个亟待解决的问题。在这项研究中，研究了受自然启发的基于包装器的元启发式算法，例如萤火虫、蝙蝠和鲸鱼优化算法，以分析适合恶意软件检测的不同 Android 权限模式。不同的机器学习分类算法，例如线性回归 (LR)、**支持向量机** (SVM)、K-**最近**邻 (KNN)、**决策**树 (DT)、**随机森林** (RF)、**梯度提升** (GB) **和极限学**习机器（ELM）

Received: August 12, 2021 / Revised: September 08, 2021 / Accepted: September 30, 2021 / Published: October 16, 2021

About the authors：Santosh Jhansi K

Corresponding author- Email:

用于评估。使用 Firefly 算法 (WFSFA) 的 Wrapper-Based Feature Selection 在特征减少方面优于其他算法，并在对具有 4115 个特征的高维 CICInvesAndMal2019 特征数据集进行试验时记录了 95.28% 的改进分类准确率。

关键词：Android 安全、恶意软件分析、静态分析、特征选择、Android 恶意软件分类、萤火虫算法 (FA)、蝙蝠算法 (BA)、鲸鱼算法 (WA)。

Introduction

In recent years, Android based mobile operating system device security [1][2] has been considered an important area of research. Unwanted permissions associated with the installation of Android applications are a major source of Android malware. For every 10 seconds, a new form of malware was discovered [3], and it was found that around 3 million malicious Android applications were discovered in 2020 alone.

A warn for the users during the application installation about suspicious permissions [4] the applications require, potentially helps the users to save from malware attacks. However, unknowingly, many users without checking the type of permission the application is asking, the users allow permission, which eventually weakens the Android permission system. In the present versions of android operating systems, a dynamic method for managing the rights increases device security. Then, the malicious applications download the malicious code after installation and run it to transfer the control to the remote server to upload the gathered data.

Several permissions are solicited by Android apps at the time of installation that knowingly or unknowingly users provide. The probability of malware induction is high in non-play store applications [5]. The application's permission mining can be a good source for identifying Android malware. The design and development of an anti-malware system that can analyze the solicited permissions by the application being installed would help Android users to be alerted for any potential threats. Dynamic analysis of Android permissions can be implemented using machine-learning tools. Android permissions are continuously evolving and have very high dimensionality. The recent and widely accepted research data CICInvesAndMal2019 [6] contains 4115 features. There is a need to experiment and find a reduced combination of features using nature-inspired optimization tools with the goal of reducing false positives and improving accuracy.

This paper presents a method for the detection of malware applications with selected permission patterns [7] and classification using ML algorithms [8] such as LR, K-NN, RF, DT, SVM, GB and ELM, by considering evolutionary computational metaheuristic algorithms such as firefly, bat, and whale optimization algorithms for improving accuracy and performance of Android Malware Detection (AMD) [9].

The rest of the paper is organized as follows: Section 2 presents the literature review, section 3 explains the background, section 4 explains the methodology, section 5 briefs about the experimental setup, section 6 details the performance analysis & experimentation results, and Section 7 gives the conclusion & future work.

Literature Review

Over the past few years, there is improvement towards the development of algorithms to reduce the dimensionality of feature sets [10]. Allix et al. [11] Investigated Android malware analysis from a forensic perspective by considering 5,00,000 benign and malware Android applications. They worked on the identification of malware through anti-viruses and the writing process of malware for Android. In addition, copying codes from blogs and tutorials makes malware writers dearth of cryptography knowledge. Because of this, both benign and malware applications share the same digital certificates, as the mitigation techniques do not check them properly. Zaman et al. [12] demonstrated a process by analyzing traffic in Android devices for malware detection in terms of behavior analysis with system calls. They created a black list of malicious applications and a log table for logging app-URLs that communicate with remote servers. They compared blacklisted domains with a log table for malware identification. Patel [13] surveyed different models for malware detection in Android systems, including DroidNative, DREBIN, ICCDetector, APK Auditor, Data Mining based Detection of Android Malware (DMDAM), AndroDialysis, and API based systems and concluded by briefing the techniques analyzed.

Wang et al. [14] addressed the problem of inefficient feature interaction by proposing a multilevel permissions extraction process. It involves association rule mining, principal component analysis, and deep cross-network techniques for feature selection and classification using different ML algorithms. The results showed that they obtained a better performance model than the state-of-the-art methodologies, with an accuracy of 95.8%. Şahın et al. [15] described a classification method for AMD in permission-based static analysis. They added weights to the features by using binary and RF techniques. Better results were observed and then classified using naive Bayes (NB). Elakkiya et al. [16] elucidated a process for Twitter spam detection using a community-based FA with fuzzy cross-entropy as the objective function for feature selection. SVM, KNN, and RF machine learning algorithms were used as classifiers. They achieved an accuracy of 90.88% with the RF classifier when compared with the traditional firefly algorithm.

Suchetha et al. [17] experimented with the feature selection process in two different stages for Twitter sentiment analysis. In the first step, the filter-based information method is applied, and then evolutionary computation-based Cuckoo Search (CS), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Firefly Algorithm (FA) are used for feature selection. In the second step, classification algorithms such as KNN, naive Bayes, and LibLinear were used for evaluation. The results indicated that the LibLinear algorithm outperformed the other. Maza and Zouache [18] developed a binary variant of the firefly algorithm compounded with a new concept for distance and attractiveness calculations. The proposed Firefly Algorithm for Feature Selection (FAFS) outperformed when compared to Particle Swarm Optimization (PSO). Li and Le [19] demonstrated a blended technique with a binary version of the bat algorithm for feature selection. Using the KNN classifier, they obtained improved results in terms of accuracy. A comparison of related studies is represented in Table 1.

Tao et al. [20] worked on a multi-label feature selection algorithm by adding a mutation

technique to the binary bat algorithm. They maximized the feature selection criterion depending on the correlation among the features. The experimental method demonstrated that there is an increase in performance when compared with the Correlation-Based Feature Selection – Genetic Algorithm (CFS-GA), Fast Multi Label Feature selection method based on information-theoretic feature ranking (FMLF), and Multivariate Mutual Information-Particle Swarm Optimization (MMI-PSO) algorithms in terms of accuracy, subset accuracy, hamming loss, and F1 score. Xu et al. [21] addressed the problem of dimensionality reduction in a large feature set. A nonlinear convergence factor is introduced by considering the PSO strategy to update the mechanism of search for prey in the WOA. The evaluated improved binary whale optimization algorithm (IBWOA) demonstrated increased performance when compared with GA, WA, and PSO.

Hussien et al. [22] described a hyperbolic tangent function in the whale optimization algorithm as a fitness measure. Both the qualitative and quantitative results indicate that the proposed binary WOA improved the accuracy along with dimensionality reduction. Hadiprakoso et al. [23] developed a hybrid dataset by combining the static and dynamic features of the Malgenome, DREBIN, and CICMalDroid datasets. They used PCA to reduce features in a combined dataset and classified them using different machine learning (ML) and Deep Learning (DL) algorithms.

S. No	Paper, Year	Dataset Used	Classifier	Feature Selection	No. of Features	Accuracy
1	[15], 2018	Own Dataset	KNN	NA	330	96%
2	[16], 2020	Twitter	RF	FA	62	90.88%
3	[17], 2019	Twitter	LR	FA	5	78.60%
4	[19], 2019	Multiple Datasets	KNN	BA	60	88.40%
5	[22], 2017	Multiple Datasets	KNN	WA	60	84.43%
6	[29], 2018	MODroid	LDA, RF, SVM	NA	134	85.5%
7	[30], 2018	Own Dataset	RF, SVM, KNN, NB, LR	NA	696	81%
8	[31], 2019	MODroid	NB, RF, SVM	NA	134	86.06%
9	[32], 2014	Google Play Store	Dalvik Virtual Machine	NA	-	85%
10	[25], 2021	CICInvesAndMal2019	DT	BA	4115	95.92

Upon their analysis observed that the GB algorithm outperformed the comparison algorithms. Amer [24] demonstrated an

ensemble model for Android malware classification using a combination of permissions. Their ensemble model using RF, MLP, Adaboost, SVM, and DT yielded better results when individual classifier results were considered. Varma et al. [25] experimented with wrapper-based bat, cuckoo search, and gray wolf optimization algorithms to reduce the features to be analyzed in a high-dimensional feature set. The results showed that the wrapper-based bat optimization algorithm with the DT classifier outperformed other competing algorithms with an accuracy of 95.92%.

This paper proposes a wrapper-based feature selection method using the firefly algorithm for recognizing the reduced subset of features in a high-dimensional dataset for Android malware classification. In addition, the performance of WFSFA is compared with Wrapper-based Feature Selection using the Bat Algorithm (WFSBAT) and Wrapper-based Feature Selection using the Whale Optimization Algorithm (WFSWOA).

2. Background

2.1. Malware Analysis

The examination of the functionality, origins, and potential consequences of a given malware sample is called malware analysis. Any computer software that contains malicious software or malware is designed to damage the host system by stealing user's information. Various categories of malware are shown in Figure 1.

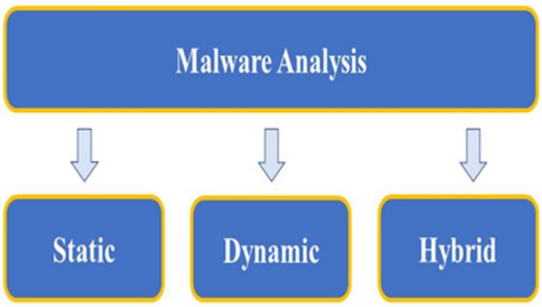

Fig.1 Types of malware analysis

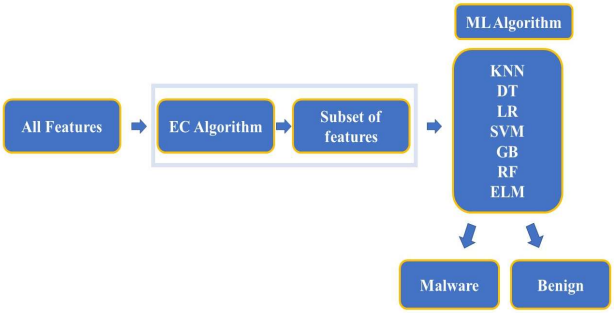

Fig.3 System architecture

The process by malware detection falls into one of two categories: static analysis, which uses a signature-based approach and dynamic analysis, which uses a behavior-based approach. In the static analysis, the malware sample is analyzed without executing it. It investigates the impacts of the system. In dynamic analysis, the malware sample is analyzed through behavior and static actions during its execution for a better analysis of the malware. The combination of both analysis results in the hybrid analysis produces better results with an increased cost.

2.2. Feature Selection

In data preprocessing, feature reduction and feature selection play an important role in choosing a suitable feature set. The feature reduction technique, also called dimensionality reduction, reduces the number of features in the feature space. Feature reduction minimizes the multi-co-linearity in the feature set. On the other

hand, the feature selection technique identifies the most influential features of the ML model.
It removes irrelevant and redundant features, which decrease the performance of the ML model. The reduced feature set improves the ML model performance by minimizing the computational cost in the model learning phase and increasing model understandability. It reduces over fitting and decreases the training time of the model. The different types of feature-reduction methods are shown in Figure 2.

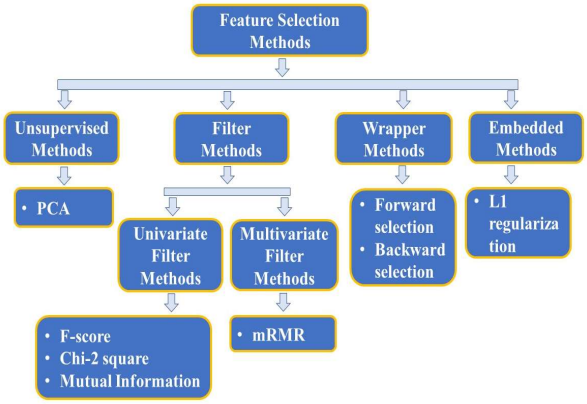

Fig.2 Feature selection methods

Each feature selection method has its own advantages and disadvantages, and it incorporates a ML model to infer the importance of a feature in ML model building. The wrapper methods are slower in deducting the influential features when compared to the filter methods. As, the number of features in the feature set is directly proportional to the computational cost of the ML model. Hence, it is not suitable for large datasets. The learning capability of the wrapper methods enables them to find the best subset of features with a small feature set.

3. Methodology

The feature selection process procures the most promising feature subset and enhances the performance of the ML model by discarding the misleading features. To examine the authoritative features in high-dimensional data, a wrapper-based metaheuristic method using firefly, bat, and whale optimization.
Initially, all features in the data were considered. A subset of features is selected using an evolutionary computational algorithm. The obtained feature subset is used for the classification of intrusion detection in Android malware using a machine learning algorithm. The architecture of the proposed model is shown in Figure 3. To find an optimal solution with an underestimate feature set, choosing the objective function plays a key role in converging the fitness during optimization.
To calculate the fitness of the selected feature subset, the number of features selected and the error obtained during the evaluation of the machine learning model are used as shown in Eq. (1)

$$f(x) = \tau \times error + (1 - \tau) \times \frac{u - s(l)}{u} \quad (1)$$

where, $\tau \in [0,1]$ represents the penalty given to error while obtaining fitness of a solution over the subset of features selected, more the τ value represents higher the penalty in choosing the subset of features. u represents length of complete set of attributes and l represents length of the solution.

3.1. Wrapper-Based Feature Selection Using Firefly Algorithm (WFSFA)

The firefly algorithm (FA), [26] inspired by firefly behavior, is an evolutionary algorithm motivated to explore solution areas based on the characteristics of the light-emitting nature of fireflies. Behavioral characteristics of fireflies:

- For any two given fireflies, the attraction between them depends on the brightness of the firefly.
- The firefly having less brightness shift towards firefly with higher brightness.

- Random movement is used, if there is no brighter firefly.

The attraction between two given fireflies is calculated using Eq. (2)

$$\beta_r = \beta_0 \times e^{-\gamma r_{jk}^2} \quad (2)$$

Where, β_0 represents attractiveness at a distance of r=0, r_jk indicates the distance between the fireflies j and k.

The distance between the fireflies at positions j and k was calculated using Eq. (3)

$$r_{jk} = \|x_j - x_k\| = \sqrt{\sum_{i=1}^{n} \left(x_{j,i} - x_{k,i}\right)^2} \quad (3)$$

Where, x_(j,i) and x_(k,i) are spatial components of ith component at jth and kth fireflies, n represents the number of dimensions.

The movement of fireflies when one firefly is attracted by another firefly is determined using Eq. (4)

$$x_j = x_j + \beta_0 \times e^{-\gamma r_{jk}^2} \times \left(x_j - x_k\right) + \alpha \times \left(random - \frac{1}{2}\right) \quad (4)$$

where, current position of firefly (j) is x_j in first term. The updation of position towards brighter firefly is defined in second term. random represents random number between 0 to 1. Firefly is moved randomly w.r.t α (mutation coefficient), when there is no brighter firefly.

Algorithm 1: WFSFA

Firefly population initialization
Define objective function f(x)
Define light absorption coefficient r.
while ($t < maxIteration$)
  for each firefly i ($\forall\, i = \{1, 2, \ldots n\}$)
    for $\forall\, j = \{1, 2, \ldots i\}$
      Obtain light intensity I_i using $f(x)$
      if ($I_i < I_j$)

$$x_j \leftarrow x_j + \beta_0 e^{-\gamma r_{jk}^2} \left(x_j - x_k\right) + \alpha \left(random - \frac{1}{2}\right)$$

      else
        Random movement of firefly *i*
      end if
      Change attractiveness with distance r and light absorption
      Determine new solution and light intensity is revised
    end for
  end for
  Rank fireflies as per their light intensities and obtain the current best
end while

3.2. Wrapper-Based Feature Selection Using Bat Algorithm (WFSBAT)

The echo location behavior and characteristics of bats to find their prey inspired the evolution of the bat algorithm [27]. Usually, bats reduce their loudness and conversely increase the rate of ultrasonic sound emission during the chase of their prey. The frequency vector, velocity vector, and position vector were used for each artificial bat in the binary bat algorithm. Either 0 or 1 represents the position of a binary bat. The velocity vector is updated using Eq. (5)

$$V_i(t+1) = V_i(t) + (X_i(t) - G_{best}) \times F_i \quad (5)$$

where, F_i, V_i and X_i represents the frequency, velocity and position vectors of ith bat. G_best represents optimal solution that has obtained. Eq. (6) is used to update the frequency of the ith bat:

$$F_i(t+1) = F_{mn} + (F_{mx} - F_{mn}) \times \beta \quad (6)$$

where, F_mx represents maximum frequency, F_mn represents minimum frequency and β represents a random number between 0 and 1. The bat position is updated using Eq. (7)

$$X_i = X_i(t) + V_i(t) \quad (7)$$

The pulse rate (r) and loudness (A) of the bat algorithm are updated using Eq. (8) & (9).

$$A_i(t+1) = \alpha \times A_i(t) \quad (8)$$

$$r_i(t+1) = r_i(0) \times [1 - \exp(-\gamma t)] \quad (9)$$

Where α and γ are constants. When there is a guarantee that the new solution improves the movement of bats toward the best solution, the pulse rate and loudness are updated. Movement of bats toward the best solution, the pulse rate and loudness are updated.

Algorithm 2: WFSBAT

Bat population initialization
Fitness calculation using objective function $f(x)$
$fitness\_min \leftarrow$ bat with minimum fitness value
$G_{best} \leftarrow$ minimum fitness value of a bat
while $(t < maxIteration)$
  $F_i(t+1) \leftarrow F_{mn} + (F_{mx} - F_{mn}) * \beta$
  $V_i(t+1) \leftarrow V_i(t) + (X_i(t) - G_{best}) * F_i$
  $t \leftarrow \frac{1}{1+e^{-v_{ij}}}$
  if $(t < random)$
    Generation of new bat solution
  end if
  if $(random > pr_i)$
    new bat solution is to be updated as G_{best}
  end if
  $new\_fitness \leftarrow$ new bat fitness is calculated using $f(x)$
  if $(prev\_fitness<new\_fitness)$ and $(random < A_i)$ then
    $X_i \leftarrow X_i(t) + V_i(t)$
    $A_i(t+1) \leftarrow \alpha * A_i(t)$
    $r_i(t+1) \leftarrow r_i(0) * [1 - \exp(-\gamma t)]$
  end if
  if $(new_{fitness} < fitness\_min)$
    $G_{best} \leftarrow new\_fitness$
  end if
end

3.3. Wrapper-Based Feature Selection Using Whale Optimization Algorithm (WFSWOA)

The whale-inspired metaheuristic optimization algorithm [28] is triggered by the mimicking behavior of humpback whales. A unique behavior seen in humpback whales, called spiral bubble-net feeding, is employed in the whale optimization algorithm (WOA). They start creating bubbles in a spiral shape from 12m down the surface of the prey and spirally swim toward the surface.

3.3.1. Encircle The Prey:

Humpback whales can locate and encircle their prey. As the optimal solution to the position of the whale is unknown, the current best candidate solution is considered as the optimal solution by the WOA algorithm. Once the best search agent is defined, other search agents try to update the status of their positions toward the best search agent using Eq. (10) & (11).

$$\vec{D} = \left|\vec{C}.\vec{X}_{best}(t) - \vec{X}(t)\right| \quad (10)$$

$$\vec{X}(t+1) = \vec{X}_{best}(t) - \vec{A} \cdot \vec{D} \quad (11)$$

where, $\vec{X}$ is position vector of the whale, t indicates current iteration, $\vec{X}_{best}$ represent the position vector of the best solution, $\vec{A}$, $\vec{C}$ are the coefficient vectors estimated using Eq. (12) & (13).

$$\overrightarrow{A} = 2 \times \overrightarrow{a} \times \overrightarrow{r1} - \overrightarrow{a}$$
(12)

$$\overrightarrow{C} = 2 \times \overrightarrow{r2}$$
(13)

Where $\overrightarrow{r1}$, $\overrightarrow{r2} \in [0, 1]$ represents the random vectors, and $\overrightarrow{a}$ component decreases linearly from 2 to 0.

3.3.2. Exploration Phase (Bubble-Net Attacking Method)

The updation of position in the spiral method is explored in this phase. The distance between the position of the prey (X, Y) and the position of the whale is calculated; then the movement of the whale to the prey is simulated using Eq. (14) & (15).

$$\overrightarrow{D^l} = |\overrightarrow{X}_{best}(t) - \overrightarrow{X}(t)|$$
(14)

$$\overrightarrow{X}(t+1) = \overrightarrow{D^l} \cdot e^{bl}.\cos(2\pi l) + \overrightarrow{X}_{best}(t)$$
(15)

where b represents a constant for logarithmic spiral shape determination.

3.3.3. Search Phase

In the whale optimization algorithm, instead of using the global best search agent for updating the search agent. A randomly chosen search agent was used, in contrast to the exploration. The mathematical model is given in Eq. (16) & (17).

$$\vec{D} = \left|\overrightarrow{C}.\overrightarrow{X}_{random}(t) - \overrightarrow{X}(t)\right|$$
(16)

$$\overrightarrow{X}(t+1) = \overrightarrow{X}_{random}(t) - \overrightarrow{A} \cdot \vec{D}$$
(17)

Where, $A \in [-1, 1]$.

Algorithm 3: WFSWOA

Whale population initialization
Fitness calculation for each agent using $f(x)$
$X_{best} \leftarrow$ agent with best fitness value
while $(t < maxIteration)$
  for each search agent $(\forall\, i = \{1, 2, \ldots n\})$
    update a, A, C, l and p
    if $(p < 0.5)$
      if $(|A| < 1)$
        $\vec{D} \leftarrow \left|\overrightarrow{C}.\overrightarrow{X}_{best}(t) - \overrightarrow{X}(t)\right|$
      else
        $\overrightarrow{X}_{random} \leftarrow [0, 1]$
        $\overrightarrow{X}(t+1) \leftarrow \overrightarrow{X}_{random}(t) - \overrightarrow{A} \cdot \vec{D}$
    end if
    else
      $\vec{D} \leftarrow \left|\overrightarrow{C}.\overrightarrow{X}_{random}(t) - \overrightarrow{X}(t)\right|$
      $\overrightarrow{X}(t+1) \leftarrow \overrightarrow{D^l} \cdot e^{bl}.\cos(2\pi l) + \overrightarrow{X}_{best}(t)$
  end for
  Verification of search agent, searching beyond search space for admitting them
  Fitness computation of each search agent
  if $best\ solution$ is found
    Update X_{best}
  $t \leftarrow t + 1$
end

4. Experimental Setup

The overall experimentation was implemented in a Windows 10 operating system (64-bit) with an Intel® Core™ i^5 CPU @ 1.80 GHz processor, 8 GB RAM, and 1 TB HDD installed with Anaconda, Python platform with supporting machine learning packages as its configuration.

The very high-dimensional Android malware dataset CICInvesAndMal2019 [6] was used to evaluate the experiment. The dataset contains 4115 Android permissions and 1594 instances, of

which 1187 samples are benign and 407 samples, are malware application types. These samples were derived from a group of 42 different malware families, which were grouped into Adware, SMS, Ransom ware, and Scare ware categories. The dataset description is provided in Table 2.

Tab.2 Description of dataset

Dataset	Features	Samples		Size
CICInvesAnd Mal2019	4115	1594		12.7 MB
		Malware	Benign	
		1187	407	

5. Performance Analysis and Experimental Setup

The evaluation metrics considered for the performance evaluation of the experimental algorithms are precision, recall, f-score, MSE, and RMSE, which are as follows:

$$\text{Precision} = \frac{True\ Positive\ (TP)}{(False\ Positive(FP) + True\ Positive)} \quad (18)$$

$$\text{Recall} = \frac{True\ Positive}{(False\ Negative(FN) + True\ Positive)} \quad (19)$$

$$\text{F1-Score} = \frac{2\times Precision \times Recall}{(Precision + Recall)} \quad (20)$$

$$\text{Accuracy} = \frac{TP + TN}{(TP + FP + FN + True\ Negative(TN)} \quad (21)$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \quad (22)$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2} \quad (23)$$

where Y_i represents the actual output, Yˆ_i represents the predicted output, and n represents the number of samples.

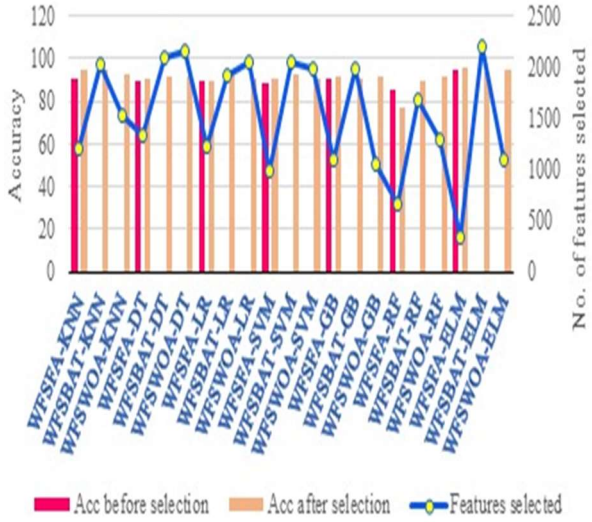

Fig.4 Performance comparison of WFSFA, WFSBAT and WFSWOA with seven different classifiers

For the classification evaluation of the experimented WFSFA, WFSBAT, and WFSWOA methods, seven classification algorithms were considered: LR, SVM, DT, GB, ELM, RF & K-NN. Various experiments were conducted to evaluate the three experimental algorithms, and the results are listed in Table 3. are attained with 50 agents and 50 iterations for each feature selection method and are graphically represented in Figure 4.

Out of all the experiments conducted, the wrapper-based feature selection using the firefly algorithm (WFSFA) obtained better performance when the ELM classifier with 500 nodes was used, in terms of precision, recall, f1-score, accuracy, MSE, and RMSE. The evaluation metrics of the WFSFA-ELM are shown in Figure 5.

Figures 6-12. represents the area under the curve (AUC) of the Receiver Operator Characteristic (ROC) curves of seven classifiers. The

AUCROC curves are generated in contrast to those in Table 3. It was observed that the area under the reduced feature set was higher. This shows that the firefly wrapper-based feature selection optimization demonstrated better performance with the ELM classifier.

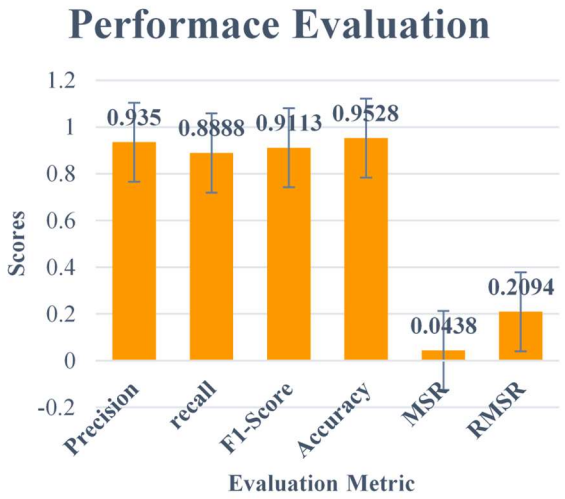

Fig.5 Evaluation metrics of the WFSFA with ELM wrapper classifier

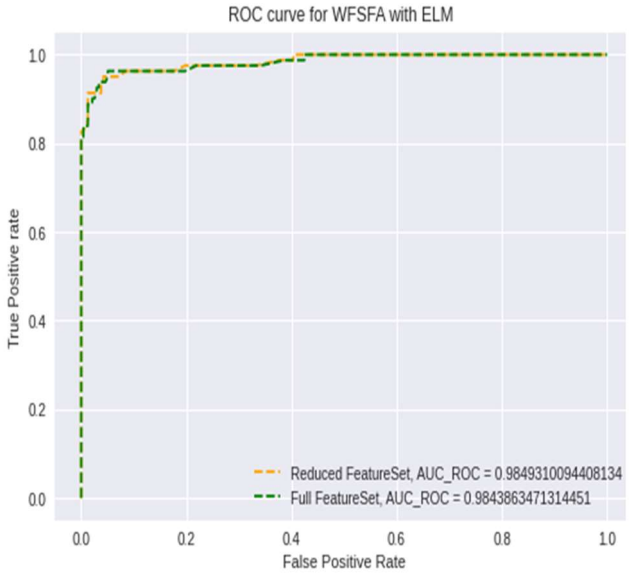

Fig.6 ROC curve for WFSFA-ELM

Tab.3 Accuracy comparison of various binary wrapper classifiers with 50 agents, 50 iterations

Exp. No	Classifier	Feature Selection Method	Accuracy before feature selection	Accuracy after feature selection	% Change in accuracy	No. of features selected	% decrease in features
		Firefly		94.6708	**3.9959**	1195	70.9599
1	KNN	Bat	90.6749	94.3573	3.6824	2030	50.6683
		Whale		92.7889	2.1140	1530	62.819
		Firefly		90.9595	0.9804	1345	67.3147
2	DT	Bat	89.9791	91.2225	1.2434	2092	49.1616
		Whale		91.2250	1.2459	2154	47.6549
		Firefly		89.3416	-0.2895	1225	70.2309
3	LR	Bat	89.6311	93.1034	3.4723	1915	53.4269
		Whale		90.9090	1.2779	2043	50.3524
		Firefly		90.2821	1.2773	998	75.7473
4	SVM	Bat	89.0048	92.4764	3.4716	2043	50.3524
		Whale		90.2821	1.2773	1989	51.6646
		Firefly		91.8495	0.9658	1094	73.4143
5	GB	Bat	90.8837	90.2821	-0.6061	1983	51.8104
		Whale		91.8495	0.9658	1062	74.1920
		Firefly		77.4294	-8.2249	667	83.7910
6	RF	Bat	85.6543	89.1316	3.4773	1688	58.9793
		Whale		91.2225	5.5682	1289	68.6756
		Firefly		**95.2897**	3.1662	**337**	**91.8104**
7	**ELM**	Bat	92.1363	94.0438	1.9122	2190	46.7801
		Whale		95.0978	2.9662	1095	73.390

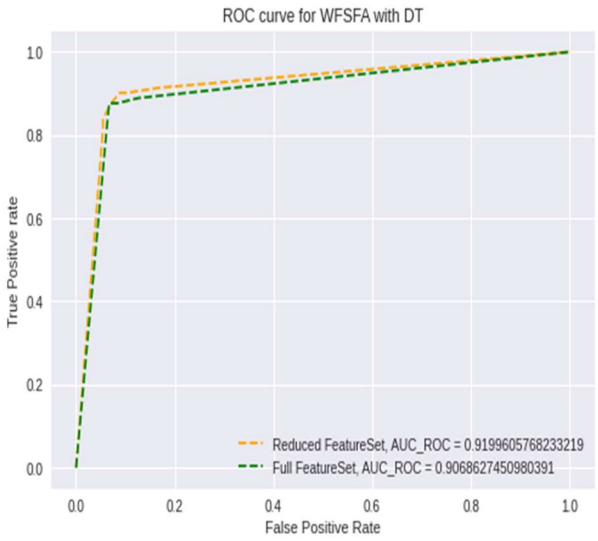

Fig. 7 ROC curve for WFSFA-DT

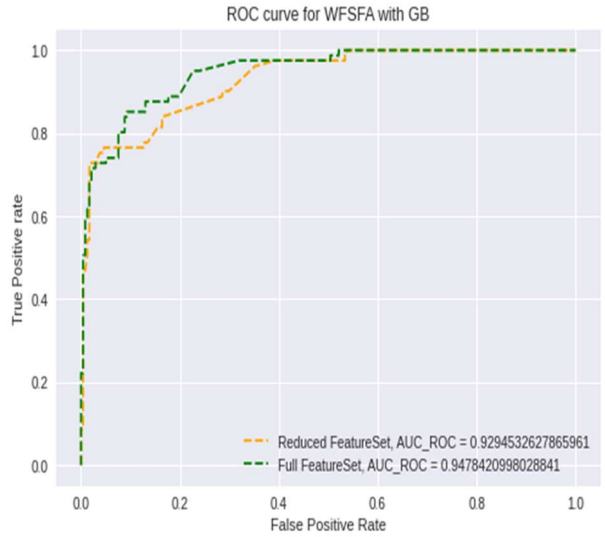

Fig. 8 ROC curve for WFSFA-GB

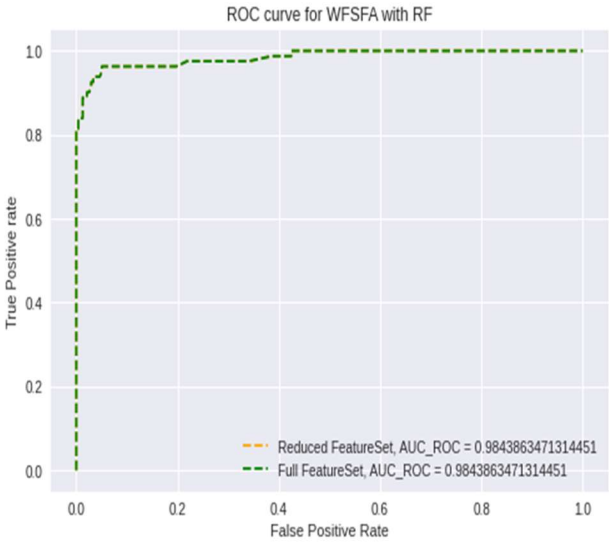

Fig.11 ROC curve for WFSFA-RF

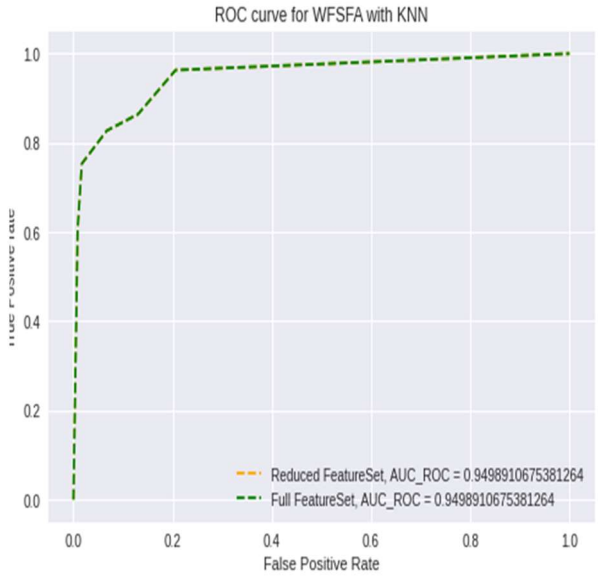

Fig. 9 ROC curve for WFSFA-KNN

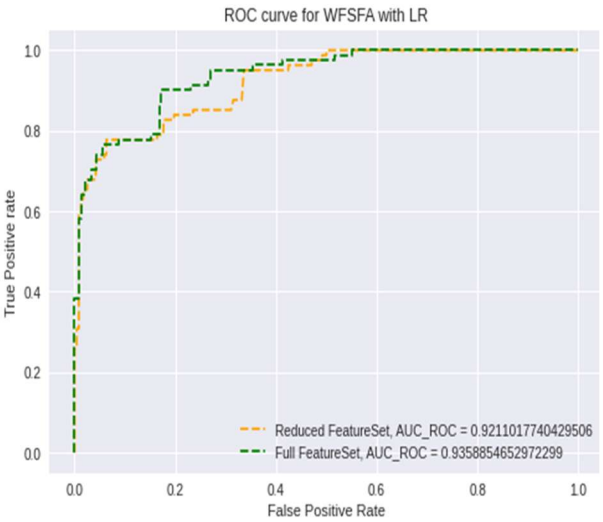

Fig.10 ROC curve for WFSFA-LR

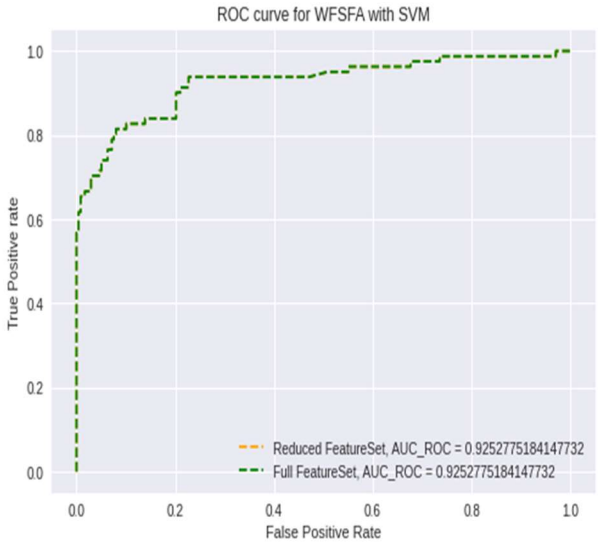

Fig.12 ROC curve for WFSFA-SVM

A list of 337 features selected by the WFSFA-ELM algorithm is listed in Table 5.

Tab.5 Selected features by WFSFA-ELM

[80, 87, 99, 132, 137, 141, 143, 163, 184, 203, 206, 225, 231, 261, 310, 316, 322, 323, 325, 357, 374, 393, 397, 432, 437, 446, 448, 452, 453, 474, 497, 512, 517, 527, 532, 541, 543, 545, 560, 566, 570, 571, 575, 585, 591, 617, 633, 649, 688, 690, 698, 731, 733, 762, 782, 829, 860, 908, 918, 919, 943, 944, 947, 948, 988, 999, 1014, 1033, 1049, 1069, 1085, 1096, 1099, 1101, 1118, 1123, 1127, 1142, 1146, 1152, 1160, 1163, 1164, 1176, 1213, 1217, 1218, 1220, 1235, 1258, 1259, 1269, 1300, 1305, 1312, 1365, 1368, 1382, 1387, 1388, 1389, 1420, 1470, 1476, 1481, 1515, 1530, 1554, 1567, 1568, 1573, 1598, 1608, 1629, 1643, 1652, 1653, 1658, 1660, 1675, 1683, 1690, 1692, 1693, 1705, 1716, 1722, 1738, 1749, 1757, 1760, 1764, 1767, 1768, 1769, 1777, 1795, 1830, 1832, 1848, 1862, 1864, 1866, 1886, 1898, 1903, 1907, 1915, 1923, 1927, 1930, 1940, 1947, 1965, 1978, 1993, 2003, 2036, 2038, 2039, 2045, 2051, 2056, 2064, 2068, 2079, 2084, 2085, 2092, 2094, 2096, 2109, 2122, 2131, 2148, 2161, 2169, 2175, 2178, 2179, 2188, 2194, 2199, 2208, 2210, 2214, 2232, 2240, 2292, 2301, 2302, 2304, 2308, 2320, 2338, 2342, 2356, 2358, 2364, 2376, 2379, 2383, 2390, 2395, 2403, 2407, 2415, 2416, 2421, 2424, 2428, 2450, 2464, 2497, 2503, 2508, 2527, 2532, 2536, 2552, 2564, 2595, 2616, 2633, 2658, 2665, 2698, 2713, 2717, 2748, 2755, 2761, 2777, 2779, 2781, 2801, 2805, 2808, 2818, 2845, 2851, 2853, 2856, 2861, 2891, 2898, 2899, 2903, 2906, 2910, 2914, 2953, 2965, 2983, 2986, 3024, 3029, 3036, 3039, 3048, 3054, 3055, 3082, 3110, 3117, 3161, 3206, 3207, 3214, 3229, 3232, 3252, 3259, 3260, 3278, 3293, 3311, 3320, 3321, 3342, 3345, 3369, 3382, 3395, 3397, 3405, 3424, 3435, 3437, 3444, 3472, 3481, 3492, 3518, 3541, 3597, 3609, 3617, 3625, 3629, 3640, 3641, 3645, 3646, 3649, 3653, 3662, 3664, 3674, 3695, 3729, 3740, 3749, 3765, 3769, 3791, 3792, 3795, 3830, 3850, 3861, 3865, 3889, 3898, 3899, 3917, 3943, 3955, 3980, 3983, 3985, 3993, 3997, 4032, 4044, 4064, 4080]

Tab.4 Comparison of similar works

Paper, Year	Classifier	Accuracy
[14], 2020	RF	90.88%
[15], 2015	Nearest Neighbor	94.37%
[21], 2017	SVM	94.49%
[17], 2019	LDA	94.64%
[22], 2018	SVM	94.72%
[25], 2021	DT	95.92% / 375
This paper	**ELM**	**95.28%/ 337**

According to the literature, there is no feature selection method using the firefly algorithm with the CICInvesAndMal2019 dataset; hence, a

comparison of similar work based on accuracies is shown in Table 4.

6. Conclusion

In light of increased smart phone usage, Android malware detection using various machine learning techniques plays a crucial role in safeguarding user data. Owing to the flexibility of adding new features to the Android operating system, the number of permissions to be validated is increasing at a high dimension. This study explores three evolutionary computational algorithms, namely, firefly, bat, and whale optimization algorithms embedded with wrapper-based classifiers. The extensively used high-dimensional data CICInvesAndMal2019 with 4115 features was employed in the experiment. Seven different classifiers, such as KNN, DT, LR, SVM, GB, RF, and ELM are used as wrapper classifiers for classification evaluation with the features selected using WFSFA, WFSBAT, and WFSWOA. The WFSWOA and WFSFA algorithms with an extreme learning machine as wrapper classifiers obtained better classification accuracy with a reduced subset of features. The WFSWOA-ELM obtained an accuracy of 95.09% with 1095 features, and the WFSFA-ELM obtained 95.28% with 337 features. The WFSFA-ELM accuracy of 95.28% was approximately equal to the accuracy of 95.92%. However, the WFSFA-ELM performed well in reducing the overall search space dimensionality by 91.81% with a chosen 337 feature subset, which is better than the 375 feature subset chosen by [25] theCICInvesAndMal2019 dataset. The results showed that the WFSFA outperformed the WFSBAT and WFSWOA algorithms when tested with 50 agents and 50 iterations on a high-dimensional feature dataset. Future work includes the design and experimentation of feature selection techniques combining both filter-and wrapper-based methods to improve the efficiency of feature selection methods. In addition, behavior analysis to monitor real-time data for Android malware classification should also be investigated.

References

[1] P.R.K.Varma, K.V.S.Raju and K.P.Raj, "Android mobile security by detecting and classification of malware based on permissions using machine learning algorithms," in International Conference on IoT in Social, Mobile, Analytics and Cloud (I-SMAC), SCAD Institute of Technology, Palladam,India, 2017.

[2] N.Anusha and B.Rajalakshmi, "Sensor based application for malware detection in android OS (Operating System) devices," in 2017 International Conference on Information Communication and Embedded Systems (ICICES), Chennai, India, 2017.

[3] https://www.techrepublic.com/article/new-android-malware-found-every-10-seconds-report-says/. [Online].

[4] S.H.Moghaddam and M.Abbaspour, "Sensitivity analysis of static features for Android malware detection," in 2014 22nd Iranian Conference on Electrical Engineering (ICEE), Tehran, Iran, 2014.

[5] W.Qing-Fei and F.Xiang, "Android Malware Detection Based on Machine Learning," in 2018 4th Annual International Conference on Network and Information Systems for Computers (ICNISC), Wuhan, China, 2018.

[6] Taheri, Laya, Kadir, Abdul, Lashkari and A.Habibi, "Extensible Android Malware Detection and Family Classification Using Network-Flows and API-Calls," in 2019

International Carnahan Conference on Security Technology (ICCST), Chennai, India, 2019.
[7] P.Xiong, X.Wang, W.Niu, T.Zhu and G.Li, "Android malware detection with contrasting permission patterns," China Communications, vol. 11, no. 8, pp. 1-14, Aug 2014.
[8] F.M.Darus, N.A.A.Salleh and A.F.Ariffin, "Android Malware Detection Using Machine Learning on Image Patterns," in 2018 Cyber Resilience Conference (CRC), Putrajaya, Malaysia, 2018.
[9] S.Arshad, M.A.Shah, A.Wahid, A.Mehmood, H. Song and H.Yu, "SAMADroid: A Novel 3-Level Hybrid Malware Detection Model for Android Operating System," vol. 6, pp. 4321-4339, 2018.
[10] P.Feng, J.Ma, C.Sun, X.Xu and Y.Ma, "A Novel Dynamic Android Malware Detection System With Ensemble Learning," IEEE Access, vol. 6, pp. 30996–31011, 2018.
[11] K.Allix, Jerome, T.F.Bissyandé, Klein, State and Y.L.Traon, "A Forensic Analysis of Android Malware -- How is Malware Written and How it Could Be Detected?," in International computer software and applications conference, Vasteras, Sweden, 2014.
[12] M.Zaman, T.Siddiqui, M.R.Amin and M.S.Hossain, "Malware detection in Android by network traffic analysis," in 2015 International Conference on Networking Systems and Security (NSysS), Dhaka, Bangladesh, 2015.
[13] Z.D.Patel, "Malware Detection in Android Operating System," in 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), Greater Noida, India, 2018.
[14] Wang, Li, Hu, Fukuda and Kong, "Multilevel Permission Extraction in Android Applications for Malware Detection," in 2019 International Conference on Computer, Information and Telecommunication Systems (CITS), Beijing, China, 2019.
[15] D.Sahın, O.Kural, S.Akleylek and E.Kilig, "New results on permission based static analysis for Android malware," in 2018 6th International Symposium on Digital Forensic and Security (ISDFS), Antalya, Turkey, 2018.
[16] E.Elakkiya, S.Selvakumar and R.L.Velusamy, "CIFAS: Community Inspired Firefly Algorithm with fuzzy cross-entropy for feature selection in Twitter Spam detection," in 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), IIT Kharagapur, India, 2020.
[17] N.K.Suchetha, A.Nikhil and P.Hrudya, "Comparing the Wrapper Feature Selection Evaluators on Twitter Sentiment Classification," in 2019 International Conference on Computational Intelligence in Data Science (ICCIDS), Gurgaon, India, 2019.
[18] Maza and Zouache, "Binary Firefly Algorithm for Feature Selection in Classification," in 2019 International Conference on Theoretical and Applicative Aspects of Computer Science (ICTAACS), Skikda, Algeria, 2019.
[19] G.Li and C.Le, "Hybrid Binary Bat Algorithm with Cross-Entropy Method for Feature Selection," in 2019 4th International Conference on Control and Robotics Engineering (ICCRE), Nanjing, China, 2019.
[20] Tao, Li and Xu, "Multi-label Feature Selection Method via Maximizing Correlation-based Criterion with Mutation Binary Bat Algorithm," in 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 2020.

[21] Xu, Fu, C.Fang, Q.Cao, S.Wei and J.Su, "An Improved Binary Whale Optimization Algorithm for Feature Selection of Network Intrusion Detection," in International Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS), Lviv, Ukraine, 2018.

[22] A.G.Hussien, A.E.Hassanien and E.H.Houssein, "A binary whale optimization algorithm with hyperbolic tangent fitness function for feature selection," in 2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS), Cairo, Egypt, 2017.

[23] R.B.Hadiprakoso, H.Kabetta and I.K.S.Buana, "Hybrid-Based Malware Analysis for Effective and Efficiency Android Malware Detection," in 2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS), pp. 8-12, Jakarta, Indonesia,2020.

[24] E.Amer, "Permission-Based Approach for Android Malware Analysis Through Ensemble-Based Voting Model," in 2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC), pp. 135-139, Cairo, Egypt, 2021.

[25] P.Ravi Kiran Varma, S Kumar Reddy Malladi, K.Santosh Jhansi and D.Pushpa Latha, "Bat Optimization Algorithm for wrapper-based feature selection and performace improvement of android malware detection,"IET networks, vol. 10, no. 3, pp. 131-140, Mar 2021.

[26] Johari, Nur, Zain, Azlan, Mustaffa, Noorfa, Udin and Amirmudin, "Firefly Algorithm for Optimization Problem," Applied Mechanics and Materials, vol. 421, pp. 512-517, Sep 2013.

[27] P.W.Tsai, J.S.Pan, B.Y.Liao, M.J.Tsai and V.Istanda, "Bat Algorithm Inspired Algorithm for Solving Numerical Optimization Problems," Applied Mechanics and Materials, Vols. 148-149, pp. 134-137, 2011.

[28] A.Lewis and S.Mirjalili, "The Whale Optimization Algorithm," Advances in Engineering Software, vol. 95, pp. 51-67, 2016.

[29] Rehman, Khan, Muhammad, Lee, Lv, Baik, P.A.Shah, K.Awan and Mehmood, "Machine learning-assisted signature and heuristic-based detection of malwares in Android devices," Computers and Electrical Engineering, vol. 69, pp. 821-841, July 2018.

[30] M.Kedziora, P.Gawin, I.Jozwiak and M.Szczepanik, "Android Malware Detection Using Machine Learning and Reverse Engineering," in CS & IT Conference Proceedings, Wroclaw University of Science and Technology, Poland, 2018.

[31] B.Wu, W.Wen and J.Li, "Android Malware Detection Method Based on frequent Pattern and Weighted Naive Bayes," in Cyber Security, Beijing, China, Springer, 2019, pp. 36-51.

[32] V. Grampurohit, "Android App Malware Detection," IIIT, Hyderabad, India, 2016.