Evaluating the Effectiveness of the Binary PSO Method in Feature Selection to Improve the Detection of Android Botnets

Peng WANG*, Zhijun WANG
College of Mathematics and Computer Science, Chifeng College, Chifeng 024000, China

***Abstract*—Android botnets endanger the security and privacy of mobile devices by doing harmful actions such as sending spam, taking data, and starting distributed denial-of-service (DDoS) attacks. Detecting Android botnets is a challenging task, as they often use sophisticated techniques to evade traditional detection methods. This paper uses the Binary PSO (BPSO) algorithm to select the important features of the Android botnet, and then adjusts the training and testing datasets accordingly, discarding the irrelevant features. Then, with the help of BPSO-SVM and BPSO-DT approaches, Android botnets are identified with high accuracy, and ten key features used to identify Android botnets are introduced. The results obtained from the approaches in question show an accuracy higher than 97% in identifying this type of malware.**

Keywords*—Android botnet; botnet; binary PSO; SVM; decision tree; BPSO-SVM; BPSO-DT

I. Introduction

The rapid development and widespread adoption of mobile devices, especially smartphones, have revolutionized the way people communicate, work, learn, and entertain. Statista reported that the global smartphone users were about 3.8 billion in 2021, and they predicted that this number would increase to 4.3 billion by 2023 [1]. Among the various operating systems for smartphones, Android is the most dominant one, with a market share of 72.2% in 2020 [2]. Android is an open-source platform that allows developers to create various applications for different purposes and users to customize their devices according to their preferences.

However, the popularity and openness of Android also make it a lucrative target for cybercriminals, who seek to exploit the vulnerabilities of the system and the applications to compromise the devices and perform malicious activities. One of the most severe and sophisticated threats facing Android users is the mobile botnet, which is a network of infected devices that can be remotely controlled by a botmaster to execute malicious commands. A mobile botnet can be used for various malicious purposes, such as stealing personal information, sending spam messages, launching distributed denial-of-service (DDoS) attacks, mining cryptocurrencies, and more. The impact of a mobile botnet can be devastating, not only for the individual users, but also for the network operators, service providers, and the society at large. For example, in 2016, the Mirai botnet infected over 600,000 IoT devices, including Android smartphones, and launched massive DDoS attacks against several websites, such as Twitter, Netflix, and Reddit, causing significant disruption and financial losses [3].

The main challenge in detecting and preventing mobile botnets is the diversity and complexity of the techniques used by the attackers [4]. Mobile botnets can employ various methods to infect devices, such as malicious applications, phishing links, drive-by downloads, and exploit kits. Moreover, mobile botnets can use different communication channels to receive commands and send data, such as SMS, HTTP, peer-to-peer (P2P), and social networks [5]. Furthermore, mobile botnets can adopt different topologies to organize the devices, such as centralized, decentralized, or hybrid. These factors make it difficult to identify and analyze the behavior and structure of mobile botnets and to design effective countermeasures against them.

The paper is structured as follows. Section II gives some background knowledge on the mobile botnet, SVM, and DT concept. Section III shows the classification and analysis of the Android mobile botnet methods based on the BPSO-SVM and BPSO-DT. Section IV presents the simulation of the results. Section V ends the paper and highlights the main results.

II. Related Works

In this section, we review the existing literature on mobile botnets based on the Android operating system. We classify the literature according to three dimensions: infection, command and control (C&C), and topology. For each dimension, we discuss the main techniques, challenges, and limitations of current research.

A. Infection

The infection dimension refers to the methods used by the attackers to compromise the Android devices and install the botnet malware on them. The infection methods can be categorized into two types: active and passive [5].

Active infection methods require the user's interaction or consent to install the malware, such as downloading and running a malicious application, clicking on a phishing link, or granting excessive permissions to a seemingly benign application. Passive infection methods do not require the user's interaction or consent, such as exploiting a vulnerability in the system or an application or using a drive-by download technique.

The majority of existing research on Android mobile botnet infection focuses on the active methods, especially the malicious applications. Several studies have proposed various techniques to detect and analyze malicious applications, such as static analysis, dynamic analysis, hybrid analysis, machine learning, and deep learning [6]. However, these techniques face several challenges, such as code obfuscation, encryption, dynamic loading, evasion, and stealthiness that make the detection and analysis of malicious applications difficult and time-consuming.

The passive infection methods are less studied in the literature, but they pose a serious threat to Android devices, as they can exploit the vulnerabilities that exist in the system or the applications, and install the malware without the user's knowledge or consent [4]. Some examples of the passive infection methods are the Stagefright exploit, which can execute an arbitrary code on the device by sending a specially crafted multimedia message, the Cloak and Dagger attack, which can perform malicious actions on the device by abusing the Android permissions system, and the Man-in-the-Disk attack, which can compromise the device by manipulating the external storage.

B. Command and Control (C&C)

The C&C dimension refers to the communication channels used by the botmaster to send commands to the bots and receive data from them. The C&C channels can be categorized into two types: centralized and decentralized [7, 8].

Centralized C&C channels rely on a single server or a group of servers to communicate with the bots. The botmaster can easily manage the bots and coordinate their activities through the centralized server. However, this also makes the botnet vulnerable to detection and disruption, as the server can be identified and blocked by the defenders. SMS, HTTP, and email are some of the centralized communication methods that the C&C server uses to control the bots [9].

Decentralized C&C channels do not rely on a single server, but use a distributed network of peers to communicate with the bots [9]. The botmaster can send commands to a subset of the bots, and the commands can propagate to the rest of the bots through the peer-to-peer network. This makes the botnet more resilient to detection and disruption, as there is no single point of failure. However, this also makes the botnet management and coordination more complex and challenging. Some examples of decentralized C&C channels are P2P, social networks, and blockchain.

C. Topology

The topology dimension refers to the structure and organization of the bots in the botnet. Topology can affect the performance, scalability, and robustness of the botnet. The topology can be categorized into three types: centralized, decentralized, and hybrid [10].

Centralized topology has a star-shaped structure, where the bots are directly connected to the central server [11]. The botmaster can manipulate the bots with ease through the server, which serves as the C&C channel. However, this topology has low scalability and robustness, as the server can be a bottleneck and a single point of failure.

Decentralized topology has a mesh-shaped structure, where the bots are connected in a peer-to-peer network. The C&C channel is a peer-to-peer network, and the botmaster can communicate with the bots via any peer [12]. This topology has high scalability and robustness, as the botnet can grow and survive without relying on a central server.

Hybrid topology has a combination of star-shaped and mesh-shaped structures, where the bots are divided into clusters, and each cluster has a leader that is connected to the central server [11]. The server and the cluster leaders are the command and control channels, and the botmaster can talk to the bots using the server or the cluster leaders. This topology has moderate scalability and robustness, as it balances the advantages and disadvantages of the centralized and decentralized topologies.

D. Support Vector Machine (SVM)

SVM is a machine learning technique that can perform classification and regression by discovering the optimal hyperplane that separates the data into distinct classes or predicts output values. SVM has many benefits, such as high precision, resistance to noise and outliers, and sparseness of the solution. However, SVM also faces some challenges, such as choosing the appropriate kernel function, dealing with large-scale and imbalanced data, and incorporating prior knowledge and domain-specific constraints. Therefore, many researchers have proposed various extensions and improvements to the standard SVM formulation, such as kernel selection, ensemble methods, fuzzy SVM, semi-supervised SVM, and constrained SVM.

Some of the recent researches that discuss these extensions and improvements are:

This research [13] reviews the existing methods for kernel selection in SVM, which can be divided into three categories: data-dependent, model-dependent, and hybrid. The study also proposes a new hybrid method that combines the advantages of data-dependent and model-dependent methods. The study assesses how well various kernel selection methods work on some benchmark datasets and demonstrates that the suggested method can attain higher accuracy and stability than the current methods.

This research [14] addresses the problem of imbalanced data classification, where the number of instances in different classes is significantly different. The study proposes a novel ensemble method that combines SVM with random subspace and bagging techniques. The study shows that the proposed method can effectively handle imbalanced data by creating diverse and balanced base classifiers and combining them with a weighted voting scheme. The study compares the proposed method with other state-of-the-art methods on several imbalanced datasets and demonstrates its superiority in terms of accuracy and robustness.

This research [15] deals with the problem of outliers, which are data points that deviate significantly from the normal distribution of the data. The study proposes a fuzzy SVM method that can handle outliers by introducing a fuzzy membership function that assigns different weights to different data points according to their degree of belonging to the classes. The study shows that the proposed method can improve the performance of SVM by reducing the influence of outliers and enhancing the generalization ability. The study tests the proposed method on several datasets with different levels of outliers and shows its effectiveness and efficiency.

E. *Decision Tree (DT)*

Decision trees are graphical models that can perform classification and regression tasks by splitting the data into smaller subsets based on some criteria. Decision trees are easy to understand, interpret, and visualize, as they mimic the human decision-making process. However, decision trees also have some drawbacks, such as overfitting, instability, sensitivity to noise, and missing values. Therefore, many researchers have proposed various techniques to improve the quality and robustness of decision trees, such as pruning, ensemble methods, fuzzy logic, and evolutionary algorithms.

Some of the recent researches that discuss these techniques are:

This research [16] reviews the existing methods for decision tree pruning, which is a technique to reduce the size and complexity of decision trees by removing unnecessary or redundant nodes. The study categorizes the pruning methods into two types: pre-pruning and post-pruning. The study also compares the advantages and disadvantages of different pruning methods and provides some guidelines for choosing the best pruning method for a given problem.

This research [17] describes the concept and applications of ensemble methods, which are techniques to combine multiple decision trees to improve the accuracy and diversity of predictions. The study covers topics such as bagging, boosting, random forests, and stacking. The study also discusses the challenges and future directions of ensemble methods for data mining.

This research [18] presents a comprehensive review of fuzzy decision trees, which are extensions of decision trees that can handle uncertainty and vagueness in the data by using fuzzy sets and fuzzy logic. The study covers topics such as fuzzy entropy, fuzzy impurity, fuzzy information gain, fuzzy splitting criteria, and fuzzy pruning. The study also compares the performance of fuzzy decision trees with crisp decision trees and other fuzzy classifiers on several benchmark datasets.

F. *Post-Quantum Cryptography (PQC)*

In recent years, post-quantum cryptography (PQC) has attracted considerable interest due to the potential risks quantum computers pose to traditional cryptographic systems. Numerous studies have delved into various aspects of PQC, focusing on algorithm development, performance enhancement, and practical application.

Liu et al. (2024) conducted an extensive survey on the performance and optimization of post-quantum cryptographic algorithms for the Internet of Things (IoT) [19]. Their research underscores the challenges and solutions in incorporating PQC into IoT devices, highlighting the necessity for lightweight and efficient algorithms to maintain security without sacrificing performance.

Another notable contribution is the survey by Ramachandran et al. (2022), which offers a comprehensive overview of lattice-based cryptographic algorithms [20]. This study examines the resilience of lattice-based methods against quantum attacks and their suitability for various cryptographic protocols. The authors also compare different lattice-based schemes, providing insights into their respective strengths and weaknesses.

In a broader scope, Jurdak et al. (2023) reviewed the current state of PQC, including an in-depth analysis of the most prevalent methods such as lattice-based, code-based, and multivariate polynomial cryptography [21]. The paper also discusses the implementation status of these methods and future research directions.

Furthermore, recent advancements in cryptographic accelerators for PQC have been documented by several researchers [19]. These studies focus on hardware implementations that can meet the computational demands of PQC algorithms, thereby enhancing their practicality for real-world applications.

Overall, the research on post-quantum cryptography is rapidly growing, with significant advancements being made in both theoretical and practical areas. Ongoing research is crucial to develop robust, efficient, and scalable cryptographic solutions capable of withstanding the emergence of quantum computing.

III. Presented Approach

In this study, we identify Android botnets and the purpose of this study is to remove inefficient features from training and testing datasets of Android botnets. Android botnets can obfuscate and encrypt the traffic sent to the botmaster, and this causes the identification of Android botnets to be associated with many challenges. Considering this issue, it can be acknowledged that the features extracted from the traffic of botnets can have ambiguous and incorrect values, which causes the wrong training of the learning model. But among the extracted features, there are several features that only by using them in machine learning approaches, the trained model can identify Android botnets with high accuracy. Therefore, the question arises as to how to identify the mentioned features from the dataset obtained from Android botnets and exclude other inefficient features from the dataset so that they can be trained in the best conditions with the help of effective features of machine learning approaches. To answer the stated question, in the next section, the Binary Particle Swarm Optimization (BPSO) approach is introduced to select key features from the Android botnet dataset.

A. *Binary PSO*

PSO is a famous evolutionary computation method, which has been used to solve many optimization problems. PSO mimics the social behavior of bird flocking, where each member (particle) is a possible solution and moves in the search space based on its own and its neighbors' best positions [22]. PSO can be divided into two main types: continuous PSO (CPSO] and binary PSO (BPSO). CPSO is designed for continuous optimization problems, where the position and velocity of each particle are real-valued vectors. BPSO is a variant of PSO for binary optimization problems, where the position and velocity of each particle are binary vectors [23, 24].

Binary optimization problems are widely encountered in various fields, such as feature selection, knapsack, scheduling, cryptography, and network design [24, 25]. In these problems, the objective is to find the optimal combination of binary variables that satisfies some constraints and maximizes or minimizes a given function. BPSO is a simple and effective method for solving binary optimization problems, as it can explore the search space efficiently and avoid being trapped in local optima [26].

The transfer function is a key component of BPSO, as it maps the continuous velocity to a binary position. The transfer function determines the probability of flipping each bit of the position vector, which affects the diversity and convergence of the swarm. Different transfer functions have different characteristics and suitability for different problems. Therefore, choosing an appropriate transfer function is crucial for the success of BPSO [27].

In general, BPSO and CPSO formulas are shown in Table I. The CPSO algorithm uses the first two formulas to change the speed and position of the particles. The BPSO algorithm changes the speed with Eq. (1) and the position of particles (binary) with Eq. (3) and (4).

In Table I, x_{id}^{t} is the location of the *i-th* particle in the d-th dimension at the *t-th* iteration, and φ_1 and φ_2 are two random numbers in a bounded domain with a uniform distribution. P_{gb}^{t} and P_{id}^{t} are the best positions discovered in the entire search space and the best position reached by the *i-th* particle at the *t-th* iteration, respectively. c_1 and c_2 are acceleration constants and ω is the inertia weight that balances the global and local searches [28]. x_{id}^{t+1} is the location of the *i-th* particle in the *d-th* dimension at the *(t + 1)-th* iteration.

The sigmoid function (S (.)) is a function of the particle's velocity in each dimension. It has a range of [0, 1]. Eq. (4) evaluates the output of this function against the outcome of Eq. (3) using a random function that produces a value in [0, 1]. Based on this comparison, Eq. (4) assigns either 1 (feature selected) or 0 (feature not selected) to each dimension of the particle ($x_{id}^{t+1}(t+1)$).

TABLE I. CPSO AND BPSO FORMULAS

Number	Name	Formula
1	*Velocity Updating*	$v_{id}^{t+1} = \omega * v_{id}^{t} + c_1 * \varphi_1 * (P_{id}^{t} - x_{id}^{t}) + c_2 * \varphi_2 * (P_{gb}^{t} - x_{id}^{t})$
2	*Position Updating*	$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1}$
3	*Sigmoid Function*	$S(v_{i,d}) = \frac{1}{1+e^{-v_{i,d}}}$
4	*Binary Position Updating*	$x_{id}^{t+1} \leftarrow \begin{cases} 0, & if\ rand > S(v_{i,d}) \\ 1, & Otherwise \end{cases}$

In Fig. 1, the BPSO algorithm performance is shown. In this approach, the mentioned technique based on Eq. (4) in Table I selects the desired features (features with a value of 1) and removes other features from the dataset (features with a value of 0).

B. BPSO-DT and BPSO-SVM

In the literature review section, the limitations of SVM and DT approaches were mentioned. In the continuation of this study, by combining these approaches with the BPSO algorithm, an attempt has been made to overcome some of the limitations expressed in these techniques.

Fig. 2 shows the flowchart of BPSO-SVM and BPSO-DT algorithms. In this form, the parameters of these methods are first initialized; then, the BPSO algorithm initializes the BPSO parameters for each particle in the search space. Next, the speed and position values of the particles are calculated. The BPSO method uses the binary position of particles to select the features of the training and testing datasets so that the optimal features can be obtained from the dataset. Then, the BPSO algorithm trains the SVM and DT methods with the Train dataset, which has the selected features. Finally, the model is applied to the test dataset (Fitness Function) and if the model accuracy is 100%, the model training is done. Otherwise, the BPSO method updates the Pbest and Gbest values based on each model's accuracy and computes the particles' speed to extract new features from the training and testing datasets. Finally, each new training dataset is given to the SVM and DT methods to train and get a new model. If any of the models can get 100% accuracy in the Fitness function, their training is done; Otherwise, the process is repeated until a certain number of iterations (for example, N times) and if it does not get 100% accuracy, it stops after the N-th iteration.

C. Fitness Function

The Fitness function is the accuracy function that evaluates the performance of the machine-learning methods. In Eq. (1), the TP and TN terms are the botnet and benign data that are classified correctly.

$$Accuracy\ (ACC) = \frac{TP+TN}{Toatal\ Test\ Sample} \qquad (1)$$

IV. EVALUATION AND RESULTS

At the beginning of this section, it should be mentioned that all approaches are implemented in Python software, and SVM and DT approaches are selected from the Scikit-Learn library available in Python.

This paper uses two Android botnets, PJapps (in EXE and BACK mode) and Geinimi (in EXE and BACK mode), from the 28-SABD databases [29], to compare the performance of the proposed methods. Also, the specifications of these datasets are mentioned in Table II. Four evaluation metrics, which are shown in Eq. (2)-(5), are used to measure the performance of each method.

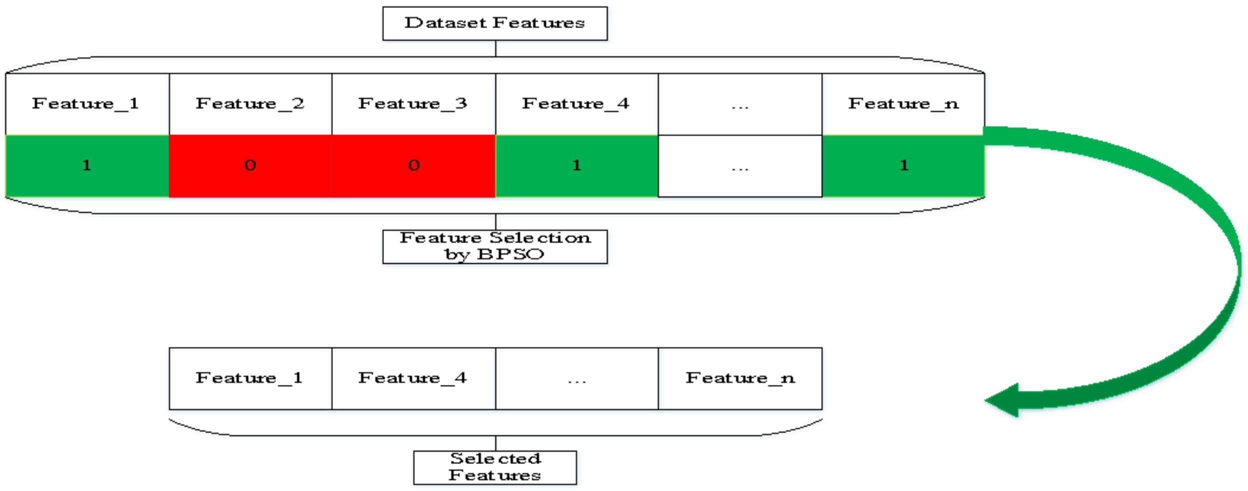

Fig. 1. How the BPSO algorithm works.

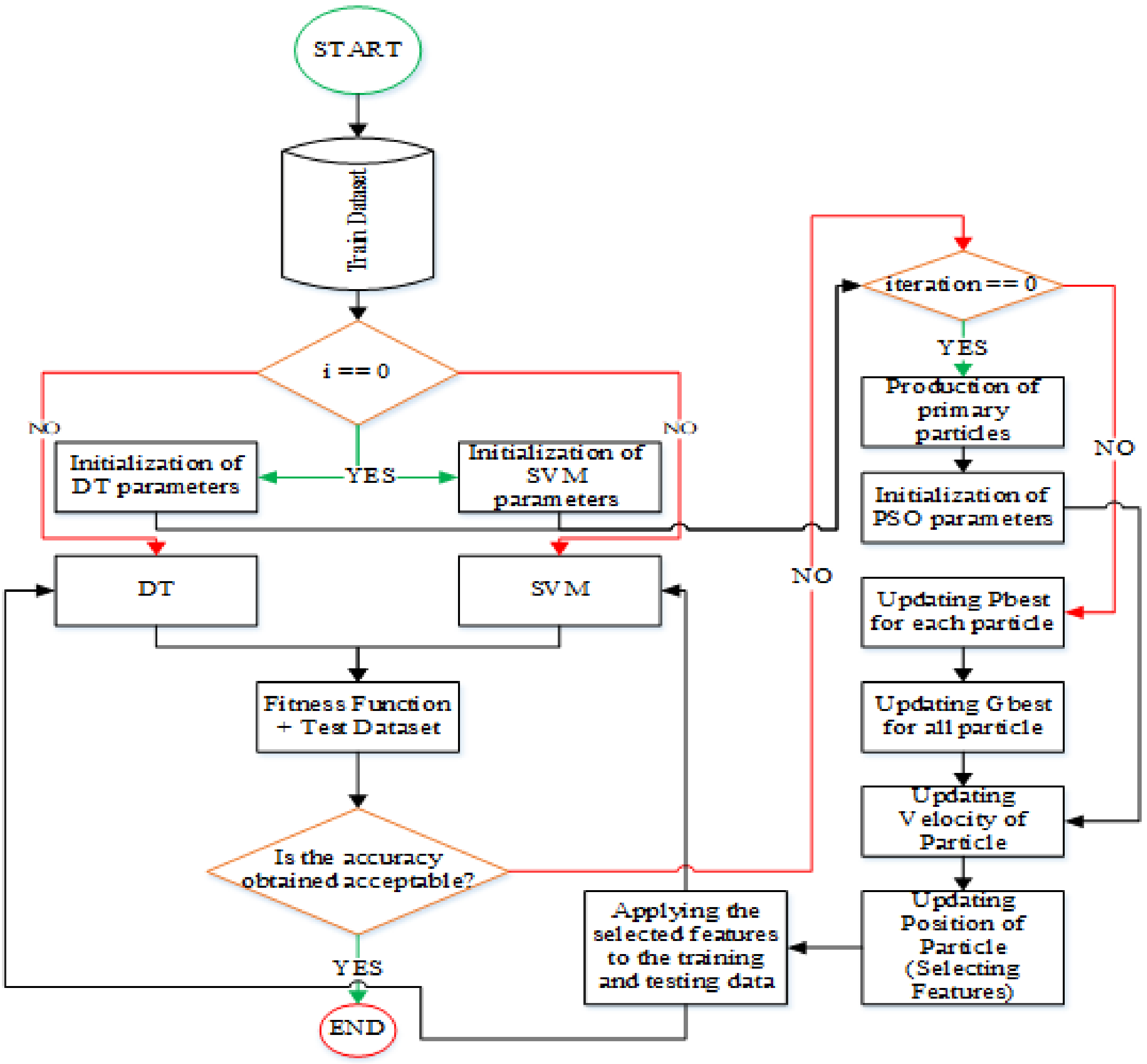
START
Train Dataset
i == 0
NO
YES
NO
Initialization of DT parameters
Initialization of SVM parameters
DT
SVM
Fitness Function + Test Dataset
Is the accuracy obtained acceptable?
YES
END
NO
Applying the selected features to the training and testing data
iteration == 0
YES
NO
Production of primary particles
Initialization of PSO parameters
Updating Pbest for each particle
Updating Gbest for all particle
Updating Velocity of Particle
Updating Position of Particle (Selecting Features)

Fig. 2. BPSO-SVM & BPSO-DT.

TABLE II. ANDROID BOTNET DATASETS [29]

Dataset	Number of Columns	Number of Rows
PJapps-Back	85	3702
PJapps EXE	85	10628
Geinimi-Back	85	4674
Geinimi-EXE	85	13757

In these equations, TP, TN, FP, and FN correspond to True Positive, True Negative, False Positive, and False Negative, respectively. Also, in Table II, the default parameters of the BPSO approach are shown.

$$\text{Accuracy(ACC)} = \frac{TP+TN}{Total\ Sample} \quad (2)$$

$$\text{Precision(Pre)} = \frac{TP}{TP+FP} \quad (3)$$

$$\text{Recall(Rec)} = \frac{TP}{TP+FN} \quad (4)$$

$$F-1 = \frac{TP}{TP+\frac{1}{2}(FP+FN)} \quad (5)$$

Before the model is trained and tested in all methods, the four datasets used are shuffled first. The techniques were applied to four different datasets and the results are shown in Tables III to VI. Based on these tables, we can say that.

TABLE III. BPSO DEFAULT VALUES

Parameter	Default Value
Number of Particles	10
Number of Iterations	10
ω	0.7
C_1	1.49445
C_2	1.49445

The techniques were applied to four different datasets and the results are shown in Tables III to VI. Based on these tables, we can say that:

1) The BPSO-SVM method performed better than the other methods on all four metrics on the PJappsExe dataset.

2) The BPSO-DT approach is the only approach that has shown better performance than other approaches in all four evaluation criteria on the PJappsBack dataset.

3) The two approaches BPSO-SVM and BPSO-DT have shown the best performance on the GeinimiExe dataset. In these approaches, all four evaluation criteria have achieved 100%. The main reason for this can be considered the selection of the best features from the aforementioned dataset. After selecting these features, the training model was trained in the best way and was able to provide the best results on the test dataset.

4) Finally, in Table VI, the BPSO-SVM approach has shown the best performance in all four measurement criteria on the GeinimiBack dataset.

As can be seen from the Tables III to VI; in all four datasets used, the BPSO-SVM approach has shown better performance than the SVM approach in all four measurement criteria. It should be noted that in some datasets, the results obtained in some measurement criteria show the performance of BPSO-SVM and SVM approaches. For example, in the PJappsBack dataset, the performance of both approaches was similar to each other in all four measurement criteria. On the other hand, the topic stated for BPSO-SVM and SVM approaches can be extended to BPSO-DT and DT techniques as well. For example, in the GeinimiBack dataset, both BPSO-DT and DT approaches have shown similar performance in all four measurement criteria.

TABLE IV. COMPARING THE PERFORMANCE OF THE SUGGESTED METHODS AND THE OTHER TWO METHODS ON THE PJAPPSEXE DATASET

Method	Dataset	Accuracy	Precision	Recall	F-1
BPSO-SVM	PJappsExe	0.9390	0.8878	0.8636	0.8755
BPSO-DT	PJappsExe	0.9051	0.9857	0.6272	0.7666
SVM	PJappsExe	0.9255	0.8666	0.8272	0.8465
DT	PJappsExe	0.6681	0.4170	0.8454	0.5585

TABLE V. COMPARING THE PERFORMANCE OF THE SUGGESTED METHODS AND THE OTHER TWO METHODS ON THE PJAPPSBACK DATASET

Method	Dataset	Accuracy	Precision	Recall	F-1
BPSO-SVM	PJappsBack	0.9663	0.5714	1.0	0.7272
BPSO-DT	PJappsBack	0.9775	0.6666	1.0	0.8
SVM	PJappsBack	0.9663	0.5714	1.0	0.7272
DT	PJappsBack	0.9213	0.3636	1.0	0.5333

TABLE VI. COMPARING THE PERFORMANCE OF THE SUGGESTED METHODS AND THE OTHER TWO METHODS ON THE GEINIMIEXE DATASET

Method	Dataset	Accuracy	Precision	Recall	F-1
BPSO-SVM	GeinimiExe	1.0	1.0	1.0	1.0
BPSO-DT	GeinimiExe	1.0	1.0	1.0	1.0
SVM	GeinimiExe	0.9980	1.0	0.75	0.8571
DT	GeinimiExe	0.9980	1.0	0.75	0.8571

TABLE VII. COMPARING THE PERFORMANCE OF THE SUGGESTED METHODS AND THE OTHER TWO METHODS ON THE GEINIMIBACK DATASET

Method	Dataset	Accuracy	Precision	Recall	F-1
BPSO-SVM	GeinimiBack	0.9854	1.0	0.4166	0.5882
BPSO-DT	GeinimiBack	0.9771	1.0	0.0833	0.1538
SVM	GeinimiBack	0.9812	1.0	0.25	0.4
DT	GeinimiBack	0.9771	1.0	0.0833	0.1538

To examine the average performance of the methods on all four datasets, we obtain Fig. 3 to 6. Based on these results, we can draw the following conclusions on the metrics used on the four Android botnet datasets:

a) The BPSO-SVM approach has shown the best performance in the "Accuracy" criterion. And the BPSO-DT approach is placed next. The key point obtained from Fig. 3 is the significant improvement in the performance of BPSO-DT compared to the DT approach in the "Accuracy" criterion.

b) The SVM approach shows the best value for the "Precision" criterion among other approaches. On the other hand, the BPSO-DT approach has been able to significantly improve the performance of the DT approach in this criterion.

c) The BPSO-SVM approach shows the best performance among other approaches in the "Recall" measure. It should be noted, that the performance of the BPSO-DT approach in this criterion has been ranked second in comparison with other approaches.

d) The BPSO-SVM and BPSO-DT approaches have shown the best performance among other approaches in the measurement criterion "F1", respectively, and have significantly improved the values of this criterion compared to the SVM and DT approaches.

The feature selection is the most significant part of Fig. 2 because the particles can get lost in the search space if they choose the wrong features. Therefore, the algorithm will not perform well if it is powerful, but the features are not relevant. Table VII shows the 10 top features of the Android botnet. These features are ranked by importance in Table VII.

In Table VIII, “Percent” indicates the percentage of a feature appearing in four different datasets. The table indicates that the most significant feature is TotalLengthofBwdPacketsfwd, which was chosen in 87.5% of the datasets. For further details on the 85 features of CICFlowMeters, refer to references [30, 31]. Android botnets communicate with the command and control server and other botnets very covertly, so applying many features does not enhance their identification but increases the FP and FN rates. Android botnet tries to avoid detection by hiding and encrypting the key features that most security researchers seek.

TABLE VIII. TEN OF THE MOST FREQUENTLY USED FEATURES AMONG THE FOUR ANDROID BOTNET DATASETS

Number	Features Name	Percent (%)
1	TotalLengthofBwdPackets	87.5
2	ECEFlagCount	87.5
3	IdleMean	87.5
4	BwdPacketLengthMin	75
5	FlowIATMax	75
6	SubflowFwdPackets	75
7	SubflowBwdPackets	75
8	ActiveMax	75
9	BwdIATMax	75
10	FwdURGFlags	75

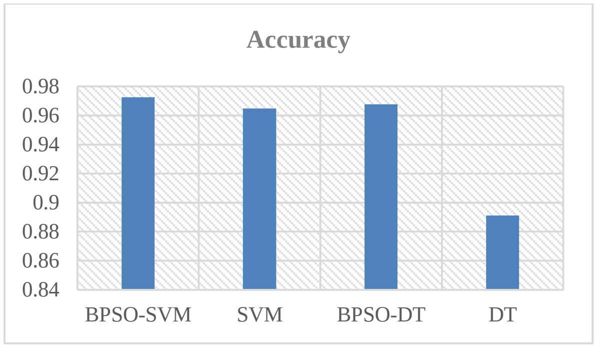

Fig. 3. Comparison of the accuracy of the four approaches used in this paper.

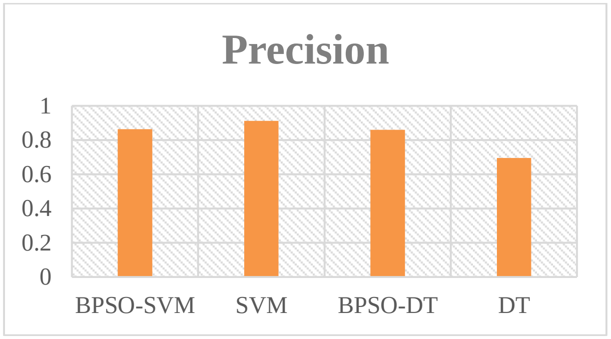

Fig. 4. Comparison of the precision of the four approaches used in this paper.

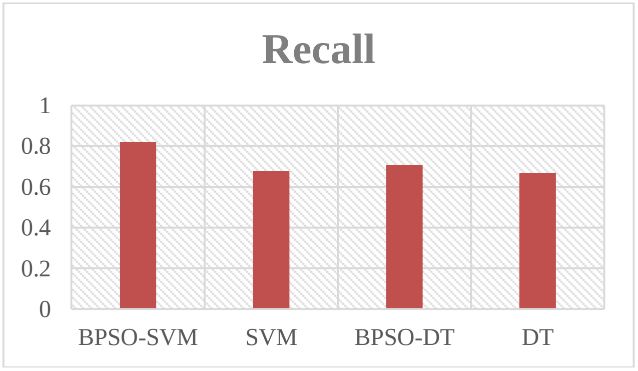

Fig. 5. Comparison of the recall of the four approaches used in this paper.

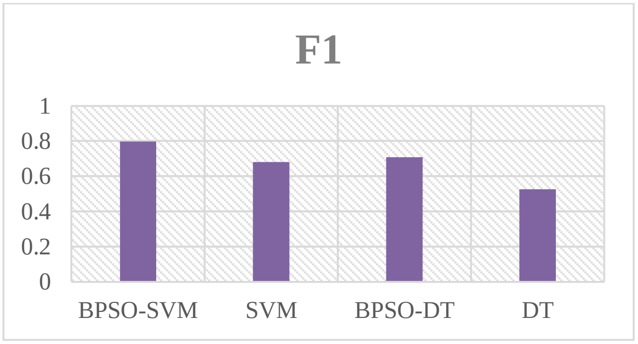

Fig. 6. Comparison of the F1 of the four approaches used in this paper.

V. Conclusion and Suggestions

Android botnets are malicious networks of compromised devices that can perform various harmful activities, such as spamming, stealing data, and launching DDoS attacks. Detecting Android botnets is a vital and difficult task, as they often use advanced techniques to evade traditional detection methods. In this study, the two machine learning approaches, SVM and DT, are used to identify Android botnets. As mentioned in this study, one of the problems of identifying Android botnets is the encryption of the traffic sent between the botmaster and botnets, which makes it impossible to identify botnets at a high rate. Some of the features from the dataset are obscure and encrypted, which hinders machine learning methods from being trained properly to detect Android botnets with high precision. This study employs the BPSO algorithm to help machine learning methods (SVM and DT) by selecting the relevant features of the dataset so that they can recognize Android botnets with high accuracy. The research results show that the best method (BPSO-SVM) has more than 97% accuracy in detecting Android botnets. Also, in this study, the top 10 most effective features that have been effective in identifying Android botnets have been mentioned.

In future research, the issue of optimizing the parameters of machine learning approaches and the effect of these parameters on the performance of these techniques will be discussed.

References

[1] Attaran M. The impact of 5G on the evolution of intelligent automation and industry digitization. Journal of ambient intelligence and humanized computing. 2023;14(5):5977-93.

[2] Adekotujo A, Odumabo A, Adedokun A, Aiyeniko O. A Comparative Study of Operating Systems: Case of Windows, UNIX, Linux, Mac, Android and iOS. International Journal of Computer Applications. 2020;176(39):16-23.

[3] Bursztein E. Inside the infamous mirai iot botnet: A retrospective analysis. Cloudflare Blog. 2020.

[4] Arora M, Skach M, Huang W, An X, Mars J, Tang L, Tullsen DM, editors. Understanding the impact of socket density in density optimized servers. 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA); 2019: IEEE.

[5] Hamzenejadi S, Ghazvini M, Hosseini S. Mobile botnet detection: a comprehensive survey. International Journal of Information Security. 2023;22(1):137-75.

[6] Arshad S, Shah MA, Khan A, Ahmed M. Android malware detection & protection: a survey. International Journal of Advanced Computer Science and Applications. 2016;7(2).

[7] Gaonkar S, Dessai NF, Costa J, Borkar A, Aswale S, Shetgaonkar P, editors. A survey on botnet detection techniques. 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE); 2020: IEEE.

[8] Shinan K, Alsubhi K, Alzahrani A, Ashraf MU. Machine learning-based botnet detection in software-defined network: a systematic review. Symmetry. 2021;13(5):866.

[9] Laabid N. Botnet command & control detection in iot networks: Itä-Suomen yliopisto; 2021.

[10] Zhou J, Xu Z, Rush AM, Yu M. Automating botnet detection with graph neural networks. arXiv preprint arXiv:200306344. 2020.

[11] Apostol I, Tica A-D, Patriciu V-V, editors. Design and implementation of a novel hybrid botnet. 2022 14th International Conference on Electronics, Computers and Artificial Intelligence (ECAI); 2022: IEEE.

[12] Dehkordi MJ, Sadeghiyan B. An effective node-removal method against P2P botnets. Computer Networks. 2020;182:107488.

[13] Awad M, Khanna R, Awad M, Khanna R. Support vector machines for classification. Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers. 2015:39-66.

[14] Hearst MA, Dumais ST, Osuna E, Platt J, Scholkopf B. Support vector machines. IEEE Intelligent Systems and their applications. 1998;13(4):18-28.

[15] Guenther N, Schonlau M. Support vector machines. The Stata Journal. 2016;16(4):917-37.

[16] Charbuty B, Abdulazeez A. Classification based on decision tree algorithm for machine learning. Journal of Applied Science and Technology Trends. 2021;2(01):20-8.

[17] Seni G, Elder J. Ensemble methods in data mining: improving accuracy through combining predictions: Morgan & Claypool Publishers; 2010.

[18] Altay A, Cinar D. Fuzzy decision trees. Fuzzy statistical decision-making: theory and applications. 2016:221-61.

[19] Liu, T., Ramachandran, G., & Jurdak, R. (2024). Post-quantum cryptography for internet of things: a survey on performance and optimization. *arXiv preprint arXiv:2401.17538*.

[20] Asif, Rameez. "Post-quantum cryptosystems for Internet-of-Things: A survey on lattice-based algorithms." *IoT* 2, no. 1 (2021): 71-91.

[21] Dam, Duc-Thuan, Thai-Ha Tran, Van-Phuc Hoang, Cong-Kha Pham, and Trong-Thuc Hoang. "A survey of post-quantum cryptography: Start of a new race." *Cryptography* 7, no. 3 (2023): 40.

[22] Nguyen BH, Xue B, Andreae P, editors. A novel binary particle swarm optimization algorithm and its applications on knapsack and feature selection problems. Intelligent and Evolutionary Systems: The 20th Asia Pacific Symposium, IES 2016, Canberra, Australia, November 2016, Proceedings; 2017: Springer.

[23] Li J, Wu Y, Fong S, Tallón-Ballesteros AJ, Yang X-s, Mohammed S, Wu F. A binary PSO-based ensemble under-sampling model for rebalancing imbalanced training data. The Journal of Supercomputing. 2022:1-36.

[24] Kennedy J, Eberhart RC, editors. A discrete binary version of the particle swarm algorithm. 1997 IEEE International conference on systems, man, and cybernetics Computational cybernetics and simulation; 1997: IEEE.

[25] Nguyen BH, Xue B, Andreae P, Zhang M. A new binary particle swarm optimization approach: Momentum and dynamic balance between exploration and exploitation. IEEE transactions on cybernetics. 2019;51(2):589-603.

[26] Fister I, Fister Jr I, Yang X-S, Brest J. A comprehensive review of firefly algorithms. Swarm and evolutionary computation. 2013;13:34-46.

[27] Zhang J, Huang D-S, Lok T-M, Lyu MR. A novel adaptive sequential niche technique for multimodal function optimization. Neurocomputing. 2006;69(16-18):2396-401.

[28] Boussaïd I, Lepagnot J, Siarry P. A survey on optimization metaheuristics. Information sciences. 2013;237:82-117.

[29] Moodi M, Ghazvini M. A new method for assigning appropriate labels to create a 28 Standard Android Botnet Dataset (28-SABD). Journal of Ambient Intelligence and Humanized Computing. 2019;10:4579-93.

[30] Lashkari AH, Gil GD, Mamun MSI, Ghorbani AA, editors. Characterization of tor traffic using time based features. International Conference on Information Systems Security and Privacy; 2017: SciTePress.

[31] Draper-Gil G, Lashkari AH, Mamun MSI, Ghorbani AA, editors. Characterization of encrypted and vpn traffic using time-related. Proceedings of the 2nd international conference on information systems security and privacy (ICISSP); 2016.