Hindawi
Journal of Electrical and Computer Engineering
Volume 2022, Article ID 3073932, 10 pages
https://doi.org/10.1155/2022/3073932

Research Article

Smart Approach for Botnet Detection Based on Network Traffic Analysis

Alaa Obeidat[1] **and Rola Yaqbeh**[2]

[1]*Basic Sciences Department, Faculty of Science, The Hashemite University, Zarqa, Jordan*
[2]*Nursing Faculty, Jordan University of Science and Technology, Irbid, Jordan*

Correspondence should be addressed to Alaa Obeidat; alaaf@hu.edu.jo

Received 31 October 2022; Revised 2 December 2022; Accepted 5 December 2022; Published 15 December 2022

Academic Editor: Yang Li

Today, botnets are the most common threat on the Internet and are used as the main attack vector against individuals and businesses. Cybercriminals have exploited botnets for many illegal activities, including click fraud, DDOS attacks, and spam production. In this article, we suggest a method for identifying the behavior of data traffic using machine learning classifiers including genetic algorithm to detect botnet activities. By categorizing behavior based on time slots, we investigate the viability of detecting botnet behavior without seeing a whole network data flow. We also evaluate the efficacy of two well-known classification methods with reference to this data. We demonstrate experimentally, using existing datasets, that it is possible to detect botnet activities with high precision.

1. Introduction

The computers store data on cloud servers. That working model enables simple access worldwide, which is essential to all online businesses and services. It offers a number of really beneficial services. Despite the fact that the Internet can be beneficial, cybercrime is also on the rise. Information security flaws, identity theft, and other threats might threaten the confidence and reliability of the information. Attackers, also referred to as "Botmasters," disseminate Trojans, malware, or both, increasing the number of existing bots on the network. Figure 1 depicts the attack mechanism of DDOS. The botnet is a network of robot computers/servers, where an attacker can control and obtain access to the systems without even knowing how it will finish.

The terms "bot" and "net" allude to robots and the Internet, respectively [1]. The attacker controls a command and control server that is used to operate the bots remotely. The control and command server (one of the bots in the botnet) is used by the attacker to direct and teach the other bots individually and collectively at the same time [2]. By infecting the network server, the botnet's size can grow. The ability of the botnet to spread allows it to do so across the Internet.

The following attacks can be carried out using botnets: phishing, click fraud, password theft, spamming, bit-coin fraud, mass identity theft, traffic sniffing, fresh malware distribution, and key logging.

The communication protocols used by botnets are Internet Relay Chat (IRC) and Hypertext Transport Protocol (HTTP) [3]. IRC is the most extensively used botnet protocol since it is very well known and is simple for a "botmaster" to utilize. IRC suffers because IRC servers are simple to locate and shut down if a botnet is located. A botnet is sometimes known as a zombie network because hacked systems are referred to as "zombie computers" or "zombie computers." The initial five years were the most important in creating the botnet. Attackers built the "Eggdrop" botnet, which they named their first, in 1993. After that, more sophisticated botnets were developed with additional features up to 2002 [4]. Most attackers during this period started using botnets, leading to a sharp rise in cyber-attacks.

Although many different machine learning algorithms have been put forth in the literature to develop various botnet detection models, almost all of these models and

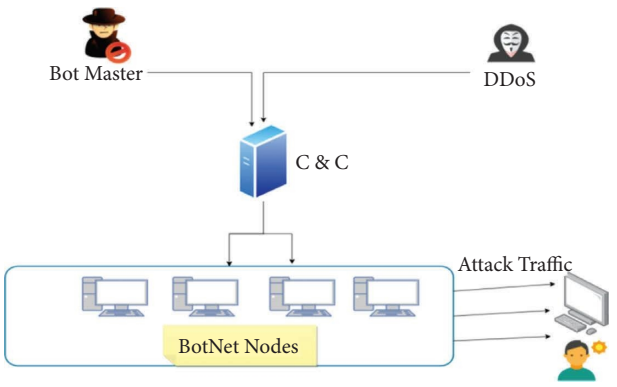

Figure 1: Generic structure of botnet.

techniques are based on extracting features (or feature development), where various feature sets are derived from the available high-dimensional datasets based on some expertise and skills [5]. On the other hand, it is discovered that the literature on botnet detection pays little attention to feature selection, which plays a crucial role in developing various machine learning models.

A botnet's life cycle is divided into five stages [6] mentioned as follows:

(1) Basic infection
(2) Secondary injection
(3) Connections
(4) Fraudulent control and command
(5) Update and maintenance

An attacker attacks the victim machine using a target system's known weakness in the first infection phase, giving the attacker more control over the target system. The attacker utilizes his newly obtained access to run additional scripts or instructions that fetch a dangerous binary from a known location during the second injection phase.

Following the installation of the malware, the victim machine runs the malicious script and turns it into a bot. The bot attempts a number of connections to the control and command server during the connection phase before actually joining the botnet after this has been established. The bots are instructed to upgrade their binaries during the maintenance phase, which is the final stage of the botnet lifecycle. This update is often done to fight against fresh attacks or enhance functionality.

Numerous currently used botnet detection methods concentrate on seeing activity from the bots either during the attacking phase or the first infection/secondary injection phase. The majority of detectors use conventional intrusion detection methods, concentrating on locating botnets based on known attack signatures by analyzing the behavior of fundamental criminal actions [7].

Our research provides a technique to identify (peer-to-peer) botnets during the attack and control and command phases. We analyze a botnet's network behavior at the TCP/UDP flow level, dividing it into several time windows and extracting attributes from them. We then utilize these attributes to categorize hostile (botnet) or benign traffic via machine learning classification algorithms. The contribution of this research is significant and unique. The technique and scale of botnet detection are the key features that distinguish this work from existing. The proposed work identifies and addresses the problem on macro level, where traffic patterns are analyzed to identify potential botnets. For this purpose, machine learning-based approach is adapted, and the genetic algorithm utilizes features to analyze the malicious activities of botnet through network traffic on IDS and DNS servers. The identification of attacks from network traffic monitoring helps us to analyze the spread of botnet binaries and the severity of attacks.

As most of the existing research is based on controlling botnet attacks on micro level, where the infected bot is identified based on its activities. Once a server is attacked by botnets, this technique enables us to observe the attacker's activity and develop numerous strategies to safeguard the network. The second method of botnet identification is based on covertly monitoring and examining network movement. The abnormal-based and DNS-based detection strategies, which have been further investigated in the paper, are the methods that come after the methods for detecting a botnet. Every DNS server and IDS server movement is monitored, and any unusual behavior is recorded to understand the attacker better and create network security measures. As opposed to other ways, this one is more effective.

2. Related Work

Botnet detection has been the subject of extensive study. This section focuses on several methods of botnet identification and suppression. The mechanisms of several botnet detection systems were explained by [8]. The research also provided network-based botnet detection methods, which involve looking at the network activity connected to the IRC protocol to detect the existence of a botnet. The botnet detection approach involves alarms when an intrusion from another network.

In previous years, IRC botnet deployment was widespread. IRC protocol's adaptability, redundancy, and scalability caused it to become widely used as a C&C method. IRC-based bots have a huge knowledge and code base, allowing their developers to reuse it to build new botnets, like the Agobot versions. Agobot's code is neatly organized and readily available online, making it simple for botnet authors to build their networks [9].

BotMiner [10] bases its detection on the collective behavior of the individual bots inside a botnet. It makes use of the fundamental consistency of botnet behavior. It attempts to watch and group identical behavior occurring concurrently on numerous network devices to discover botnets. In order to initially group network traffic characteristics that are similar, BotMiner uses "C-plane" clustering. To enhance performance flows with guaranteed safe signatures (such as those for some well-known protocols) are analyzed out of their list. After finding similar flows, BotMiner employs a

second "A-Plane" clustering method, which classifies flows according to the activities they demonstrate using anomaly detection through Snort.

Based on the progressive discrete Fourier transforms, the authors of [11]suggested a data mining-based method to identify botnet that achieved a detection performance of 99% with a rate of false positives of 14%. In their study, we record network flows and transform them into feature streams containing attributes like the flow's duration and the number of packets exchanged. The DFT is then used to increase performance by minimizing the size of the problem by calculating the distance measure of the first few coefficients of the transform [12]. The authors first group these characteristic streams using a clustering approach. Singular bots inside the same botnet may have similar flow patterns. Therefore, pairs of flows and related hosts are labeled as a suspect based on observation. A typical rule-based detecting method may then be able to verify the suspicion [13].

In [14], a single objective genetic algorithm is employed to explore the large space of candidate attributes to identify the optimal subset that could boost the effectiveness of the GAs-based botnet detection method. The authors of [15] describe the C4.5 algorithm-based botnet flow classification system. Consistency Subset Evaluation's greedy search mechanism is used to deterministically select a set of characteristics from the input data packet capture (PCAP) file. The classifier algorithm organizes groups of similar flow traffic to obtain the behavior pattern from each flow [16]. In addition to analyzing P2P botnets, the proposed system can also analyze HTTP botnets by extracting patterns from the botnets and applying them at an application layer [17].

The challenge of botnet detection and network security is currently addressed with the help of machine learning-based techniques. In [18], the robust Cubature Kalman Filter is utilized for network state estimation. As the problem of botnet attack is dynamic and complex, the adaptation of the defense mechanism depends on the accuracy type and state of the attack. The Kalman filter and its variant are tested on simulations of the IEEE 9 and New England 16 machines bus systems.

Similar to this research, false attack detection methods based on deep learning are used to protect networks from attacks [19]. The feed-forward DCNN model analyzes the data of all nodes while preserving data security and applying a federated attack detection schema. This dl model is tested on simulation of the IEEE-14 and 118 bus systems.

In [20], an active detection technique for FDIAs based on a GRU-CNN hybrid neural network is developed. It combines the benefits of gate recurrent unit (GRU) and CNN in regards to the temporal memory and feature-expression ability. Taking into account the distinctive limits of the parallel mode, an active and passive hybrid detection approach for FDIAs is constructed using the outcomes of combined knowledge-driven and data-driven parallel detection.

3. Methodology

3.1. Genetic Algorithm. The core principle of the class of flexible search algorithms known as "genetic algorithms" (GA) is based on a group of operations. GA begins with initializing a population made up of several people. Then, people are assessed using the fitness function for the specified task. The entire population is then subjected to the regenerating, crossover, and mutation operations to produce new and ideally improved individuals. The literature offers several selections, mutations, and crossover operations. Until a termination condition is justified, the process of creating new generations by selecting, crossover, and mutation operations will continue.

3.2. Network Traffic Analysis. Traffic analysis is a contemporary method that aims to address some drawbacks of payload inspection. Traffic analysis takes advantage of the fact that bots in a botnet frequently exhibit uniform traffic behavior, distinctive exhibit information, and communication behavior [21]. These behavior patterns can be described and classified using a group of features that set them apart from non-malicious traffic and techniques. Since the traffic of data analysis does not rely on the contents of the packets, it is unaffected by encryption. Dedicated equipment can extract this data with great performance and minimal network impact.

Typical traffic analysis-based detection systems examine the majority of network traffic between any two nodes. This strategy is workable for offline detection but is ineffective for real-time botnet behavior identification. The duration of a network flow between two nodes might range from seconds to more than one day, and in many cases, it is preferable to identify a botnet attack as soon as possible [22, 23].

This research describes a traffic analysis-based detection method that enables real-time botnet activity detection by briefly analyzing these data flows' features. We take advantage of certain characteristics of botnet traffic to carry out this detection with a high degree of confidence, even when there is non-malicious traffic on the network [24].

Our detection framework comprises two phases during the operation. In order to train our classifiers to recognize the two classes of data, we give our detectors a set of known criminal and noncriminal data attributes during the training phase [25].

When everything is finished, the system enters the detection stage, where it actively monitors network traffic while classifying the attribute vectors produced by active flows. The flows mentioned above may be marked as suspicious if a particular set of attributes has been identified as "malicious" in the live data.

3.3. Model of the Proposed Method. Figure 2 presents the different stages in the proposed methodology that exploits the network data traffic to detect the botnet attacks early before taking complete control of the system and becoming a victim. The different steps of the model can be described as follows:

(i) Network traffic: Network traffic can be tested directly using the data from the network card directly or from a database that includes different data packet types (normal and abnormal) [26].

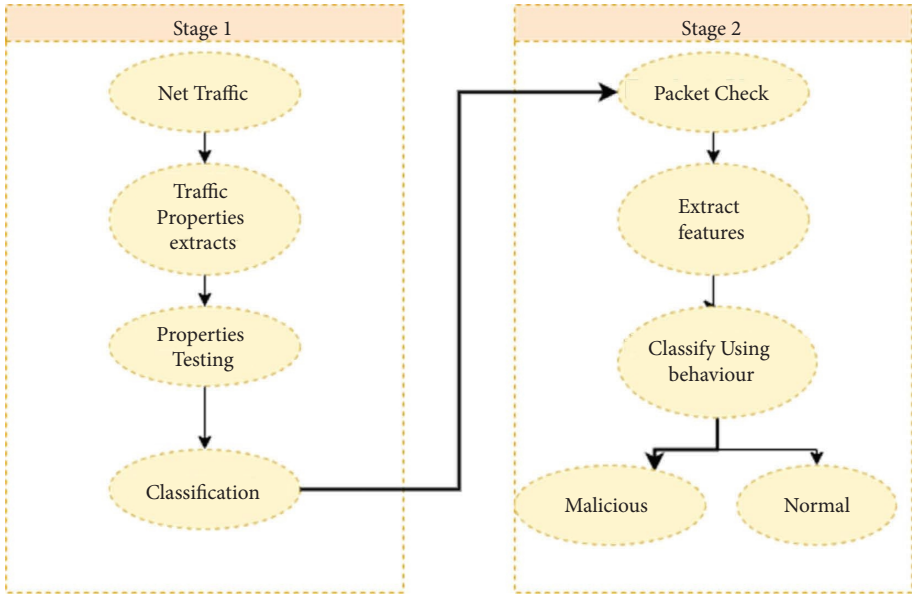

Figure 2: The proposed model architecture.

(ii) Traffic properties extract: In this research, we use the NTA IDS tool to extract the most important features of the network and packets based on a group of rules and procedures to separate the relevant features we need in this study, but we focused mainly on the data packet features that include (header, body, and trailer).

(iii) Properties testing (training): The features will be saved in a CSV file after the appropriate data has been extracted from the NTA IDS log files. The random forest algorithm was used to accomplish this task [27].

(iv) Packet check: We investigate the abnormal and normal traffic rate and packet size that can indicate that the data is normal or could be a botnet attack; the method we suggest uses many factors that can lead to specifying any potential attack.

3.4. The Features of the Networks Used to Detect Attacks. The features shown in Table 1 can be defined as follows:

(i) #In packet: how many received packets in a specific time.

(ii) #Out packet: how many sent packets are in a specific time.

(iii) #Packet/Time: ratio of the number of packets (sent/received) in a specific time.

(iv) Ration (in/out) packets: percentage between income and outcome packets.

(v) #Complete connection: number of successful connections.

(vi) #Packet received from the same IP: the number of packets received from the same IP address within a specific time, and this is one of the most important features that could be used as good indicators of any potential attack.

(vii) #Same size received packets: the number of packets of the same size received.

(viii) SYN packets/TCP packets: the ratio between SYN packets and TCP packets that come to the system during a specific period.

Table 1: Properties used to detect attacks.

Feature	Assigned value type
#In packet	Number
#Out packet	Number
#Packet/time	Number
Ration (in/out) packets	Number
#Bytes in a packet	Number
Complete connection	Bool
#Complete connection	Number
#Packet received of the same IP	Number
#Same size received packets	Number
SYN packets/TCP packets	Number

The genetic algorithm-based feature learning and network traffic analysis is a dynamic schema, where the computational complexity and accuracy of the classification model are directly affected by the initial parameters. In the first stage, where the Genetic algorithm is utilized for feature learning on a given set of infectious and noninfectious (abnormal and normal) packets, the number of packets (% of the population) considered for feature learning determines the accuracy in the second stage and regularization of the technique in more complex real-time attack scenarios. Thus, choosing the initial parameters is critical.

A balanced combination of infectious and non-infectious packets in the training data set at the first stage is required to avoid the biased learning of classification

algorithms. As in real life, the percentage of infectious packets is less as compared to the non-infectious packets. This situation is similar to the credit card fraud detection dataset problem, where fraudulent activities are rare and are undersampled; thus, the trained classification model has an inherent bias towards oversampled class.

The scalability of the proposed technique is subjected to the size of the network and network traffic density. As the proposed technique works at macro level and analyzes the behavior each packet transferbetween network nodes.

3.5. System Classifier. As seen in Figure 3, the proposed classifier is responsible for separating the network traffic data to isolate IRC & HTTP traffic from other traffic and send them to the analyzer. We can examine each packet's contents and attempt to verify the information against a collection of user-defined strings in order to identify IRC traffic. In this step, we can find some strings in the first few bytes that could indicate any potential attacks; these strings like "PW" for password, "join" for joining the targeted system, "Atc" for the attack, "FK" for fake connection and much other strings that the data packets contain.

In the next step, we classify the IP addresses received in the system using IP analysis software (NetFlow analyzer) to detect the most repetitive IP addresses that send and receive messages with the system, then send the classified results to the analyzer; if there is no suspicious activity detected, the result will be stored in the database, and every time the same IP address is detected, the data recorder in the database will be incremented.

After finishing all the stages included in Figure 3, the result could be normal traffic. The traffic details will be recorded in the database, or the system will report this to the administration if there is a potential harmful attack.

3.6. Monitoring of Network Data Traffic. When hosts (bots) launch an attack, traffic monitoring is responsible for identifying hosts that are probably a part of botnet by examining the characteristics of flows and looking for patterns between them. As a result, we are recording each network flow's unique information and capturing it. For monitoring flows and recording the data we want for this section, we are employing the open-source Audit Record Generation and Utilization System (ARGUS) [4]. The following details are included in each flow record: the source IP, the destination IP, the source port (SPORT), the destination port, the duration, the protocol, the number of packets, and the number of established connections for each IP address (#EC).

The bots that are a part of the same botnet share the same traits, as was previously mentioned. When they wish to update their directives from their "Botmasters" or aim to attack a target, their comparable behaviors are more visible. They also share similar behaviors and patterns of communication. Therefore, the next stage is to find database record groups with the same connection and data packet features.

3.7. Experiments and Testing. Forty-one features represent the factors used in a computer system network in the KDD Cup 99 dataset. It takes a lot of time and involves extensive processing to analyze these factors. As a result, our research concentrates on the six different attack kinds and the eight most crucial aspects. The KDD dataset contains 284,948 connection records, of which 10% were chosen as testing records using a predetermined probability value. Two categories of attacks we take into consideration in this study: (DoS) denial of service and phishing attacks.

As depicted in (Figure 4), the main step in the proposed method is to extract the main features of the data packets received in the network system; every time the data packet comes to the system, the IP address is checked and stored in the DB shown in Table 2 above. If any keyword is detected in the packet from the predefined keywords that indicate any potential botnet attack, the flag for this IP address is checked to report that the connection must be checked carefully and take the appropriate action against it.

One of the most important features that we used to detect the attacks is the ratio between complete three-way connections TCP and the only SYN requests connections, and this is calculated using the following formula:

$$\text{Connection Percentage} = \frac{S3w\text{TCP}}{\text{SYN Req}}. \tag{1}$$

S3wTCP refers to the number of successful three ways TCP connections, and (SYN REQ) refers to the number of SYN TCP requests only. The value of this equation is between (1, 0); if the value is 1 or close to it, this will refer to the system's status as safe. Otherwise, if the value closes to zero means, the system may contain some threats, and protection action must be taken.

Another issue we take into consideration is the packet size; the normal packet size for a TCP connection may differ from one session to another for the same IP address or from one IP address to another, but if the same IP address has the same packet size in the TCP connections established at a different time, this means that the IP address could be an attacker, and the flag in the database must be checked, and urgent action must be taken. Equation (2) shows how the average packet size is computed and compared.

$$P\,\text{Savg} = \frac{\sum_{n}^{i=1} P\,\text{size}_i}{n}, \tag{2}$$

where (PSavg) refers to packet size average, $P\text{size}_i$ refers to packet size for a specific number of packets from $(1 \ldots\ldots n)$ received at a specific time. The average is computed every time for any IP address, and if the average equals the size of the incoming packet each time and there is no difference in the size, it means the packet was received from this IP, so this means that the IP address may be considered as a malicious attacker, and the flag in the database must be checked and urgent action must be taken.

Figure 5 shows the main stages in the TCP analysis process; to distinguish the normal packet from the abnormal packet, we take the three most important features as shown in Figure 6 as follows:

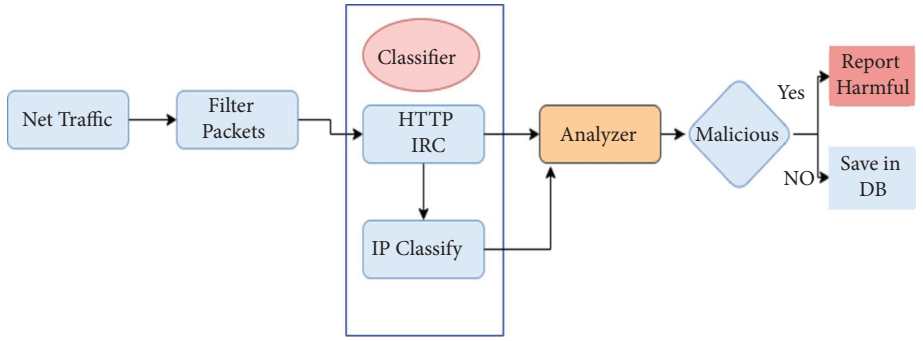

Figure 3: Architecture of the classifier.

Step 1: *Analyze the received data packet*
Step 2: *Check the IP address*
Step 3: *Increase the counter for the same IP recorded in DB*
Step 4: *Extract the common phrases in the packet body and analyze it*
Step 5: *Record the predefine phrase "malicious" in the record of IP in DB*
Step 6: *Record the type of connection (complete three-way handshake)*
Step 7: *Record the type of connection (SYN request)*
Step 8: *Record the type of connection (TCP SYN-ACK)*
Step 9: *Check the packet size*
Step 10: *Send all the recorded data in DB to the analyzer*
Step 11: *Repeat the steps for any incoming packet*

Figure 4: Pseudo code of the proposed algorithm.

Table 2: Database of recorded information for each flow.

Flow	SIP	DIP	SP	DP	Duration	#Packets	#EC	Flag
Flow 1								
Flow 2								
. . .. . .								
. . .. . .								
Flow N								

(i) SYN req > SYN-ACK: if yes, it is a good sign for the attack.

(ii) Psize = Pavg: means if all the time the packet size of the same IP address equals the average of all received packets for the same IP, this is also a good sign for something not safe in the communication with this address, as depicted in (Figure 6) there is a big difference the packet size between normal Internet packets and botnet packets.

(iii) #IP Packets/time > AllPackavg: if the number of the received packets from the same IP address is greater than the average number of the received packets from the IP address within a specific period of time, this is also a good indicator for a botnet attack.

All the information gathered after the execution and this stage will be recorded in the system database and updated every time for machine learning to monitor the system's status and control and block any potential attack in the early stages before taking full control of the system.

3.8. Performance Evaluation. To evaluate the proposed botnet attack method, we have developed a discrete-event custom-built simulator using Python. The simulation incorporates a various number of connections. Moreover, we designed a separate class of packets where each packet has a source address, a destination address, and the other important features mentioned above. A total of 2,000 to 4,000 packets are created for each evaluation, which sums up to a total amount of more than 50,000 packets. Nevertheless, our simulation performs extensive calculations to compute the arguments we need to judge whether the packet is normal.

As depicted in Figure 7, the proposed algorithm can detect the largest number of botnet packets. Nevertheless, the proposed scheme performs well with an increase in the number of packets. Figure 6 highlights the relationship between the number of connections in the network and the detection schemes' performance. It is quite evident that the proposed method performs well, disregarding any other metrics. The detection rate would reach 97% even if the number of evaluated connections increased obviously. The ratio can be computed using the following formula:

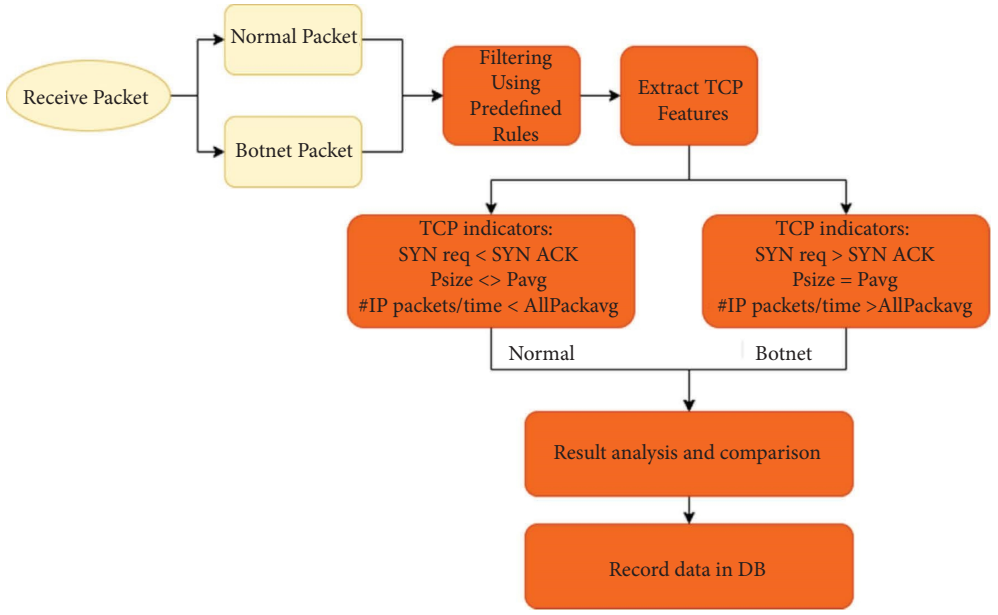

Figure 5: TCP analysis.

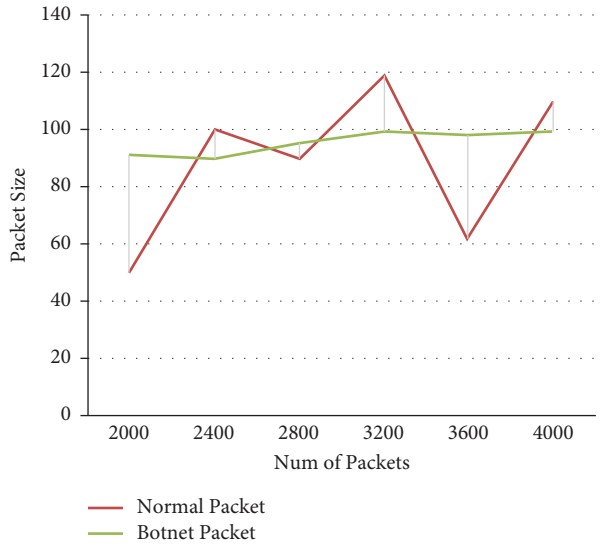

Figure 6: Normal packet size vs botnet packet size.

$$D_{\text{ratio}} = \frac{\sum \text{DBC}}{\sum \text{All Con} + \sum \text{Normconn}}, \tag{3}$$

where D_{RATIO} refers to the botnet detection ratio, DBC refers to detected botnet connections, (Allcon) means all the connections evaluated using the simulator, and (Normconn) refers to the normal connection in the evaluated dataset, where the connections used in the dataset are shown in Table 3.

3.9. Performance Measurements. We conducted a number of experiments to examine the performance of the proposed method to determine the best performance that was feasible for each argument. The performance of the simulation results is evaluated using the following metrics: accuracy, recall, and precision. The metrics are defined as follows (considering the botnet class as positive), where TP denotes true positives, FP denotes false positives, FN denotes false negatives, and TN denotes true negatives; these measurements are defined as follows:

(i) Accuracy: The percentage of events that were successfully predicted compared to the total of all predictions:

$$\frac{TP + TN}{TP + TN + FP + FN}. \tag{4}$$

(ii) Precision: All true positives divided by all positive predictions, presented as follows::

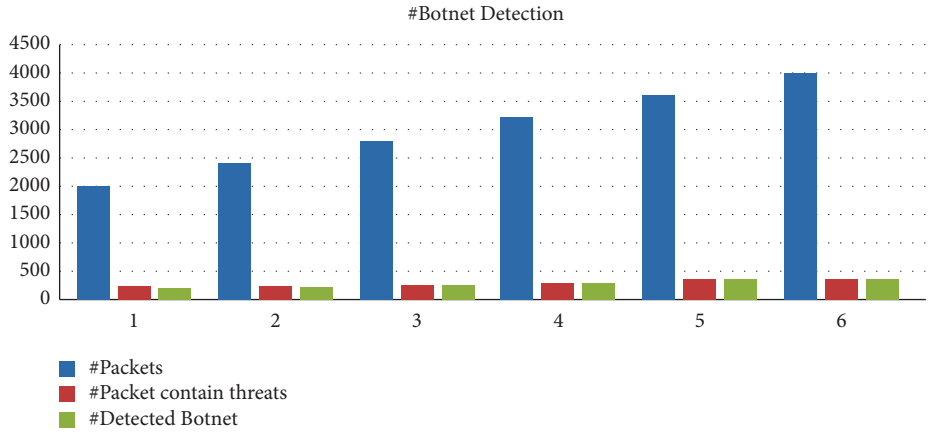

Figure 7: Botnet detection results.

Table 3: Connections used in simulation from the dataset.

#Packets	#Packet contains threats	#Detected botnet
2000	220	213
2400	250	241
2800	280	277
3200	310	302
3600	350	342
4000	390	381

Table 4: Results from executing the simulator for 6 iterations.

	Accuracy (%)	Precision (%)	Recall (%)
Iteration 1	97.2	97.1	96.8
Iteration 2	97.3	97.2	97.1
Iteration 3	98.2	98.1	97.9
Iteration 4	97.8	97.6	97.3
Iteration 5	97.6	97.4	97.1
Iteration 6	98.2	98.0	97.9

$$\frac{TP}{TP + FP}. \tag{5}$$

(iii) Recall: true positives divided by positive results. This pattern indicates that out of all potential positives, how many were found by the model?

$$\frac{TP}{TP + FN}. \tag{6}$$

We can show that the botnet training and detecting model produces effective results through the experimental findings in Table 4. This demonstrates that the suggested methodology and the traits chosen and derived in the research have produced positive results. Remarkably, the experimental portion of detecting fraudulent packets is 100 % correct, demonstrating that the training-developed model generated a model for detection. The features that we picked and proposed characterize anomalous connection behaviors. As a result, it is clear from the experimental findings in this study that they precisely distinguish between botnet and normal connections.

3.10. Tuning Parameters of the Proposed Method. Random search algorithms built on evolutionary concepts are known as genetic algorithms (GAs). Numerous NP-hard evolutionary computation problems have been solved using GAs. Appropriate parameter values are required for effective evolutionary algorithms (EA), and GA is no exception. However, in reality, parameters are typically chosen based on the implementer's experience and traditions, which favor crossover over mutation. The parameters are provided in Figure 8 as follows:

(1) The settings of factors like the mutation rate, crossover rate, population size, and elitism percentage (parent rate) have a significant impact on a GA's success. The parameter tuning issue is the challenge of anticipating good parameter values.

(2) The algorithm may employ various parameter values at various points to arrive at (near-) optimum solutions. The parameter control problem is the method by which this change is managed.

(3) Different approaches of coding the same GA algorithm can have a significant impact on performance. For instance, whereas some GAs are easy to develop in parallel, others are more challenging. Genome coding, another fixed feature that might have a big impact, is a design parameter as well. In addition, things like objective selection and termination criteria may have an effect.

Therefore, for the proposed approach, we take into consideration the following parameters when running the simulator from one iteration to another to find the optimal detection rate:

(i) Population size

(ii) Number of generations

(iii) Fitness function

3.11. Limitation. Existing traffic-based botnet detection techniques described in the literature have some significant drawbacks, much like botnet detection techniques that rely

Algorithm 2 : Genetic Algorithm for Hyperparameter Tuning

```
Result: The fittest hyperparameter in population
populations⟵[list of n models with different hyperparameter];
generation ⟵ 0;
while generation<max generation do
    train_and_evaluate (population);
    new_gen ⟵ retain the m fittest individuals;
    new_gen ⟵ append random individuals to promote diversity;
    mutate (new_gen);
    new_gen ⟵ append offsprings through crossover until k;
    population ⟵ new_gen;
    generation ⟵ generation+1
end
```

Figure 8: Parameters tuning of the proposed method.

Table 5: Effectiveness comparison with other detection methods.

	BPNN (%)	GA (%)	GNN (%)	The proposed method (%)
Detection rate	90.3	93.4	95.7	97.5
False negative	9.7	6.6	4.3	2.1

on statistical properties of packet traffic or deep packet inspection. Many of them use a simulated environment to implement the botnet detection scheme. If bots perform differently from the expected norm, this strategy could produce unintended results. The amount of data that needs to be examined to detect botnets is still relatively big, even though many of the traffic-based detection techniques use filtering to exclude bot-free data before applying a detection approach. In addition, if the dataset is vast, the computational cost for the detection strategy is frequently considerable, which is a significant drawback if faster detection is needed. In the proposed method, still we have to investigate the data packets, but it is a rule-based technique using a set of rules that can be followed easily and not consumes a lot of cost in terms of time and space complexity.

We can see in Table 5 that the detection rate of the proposed method is 97.5% and the rate of false report is 2.1%, the identification rate of the mixed genetic algorithm and neural network GNN is 95.7% and the false report is 4.3%. The detection rate of the genetic algorithm is 93.4% and the false report is 6.6% and BPNN is 90.3% and 9.7 respectively. As we can see the proposed method in this study is superior to other methods in detecting botnet attacks.

4. Conclusion

This research examines the nature of botnets and the dangers they pose to computer networking systems. In this study, we have covered a new method for detecting and keeping an eye on botnets as well as the network trapdoors that botnets exploit to attack servers. The article concludes by stating that everyone in the globe now uses the Internet as a basic necessity and that it is home to a variety of works. Every time they break in and utilize botnets to steal data, cybercriminals use new tactics. To protect our server and to detect and eliminate the botnet from it without harming our data, new approaches must be developed. Based on the results, it can be concluded that the suggested method is an effective mechanism for enhancing the security of computer networks since it can identify any connections that lead to breaches into a network. In the future, we can use a multiple objectives scalable algorithm, such as using a mixed algorithm of GA, GNN, and traffic data analysis to increase the detection rate of the botnet and maximize the precision rate and minimize the false reports of the algorithm to reach a 100% level of accuracy.

Data Availability

The datasets generated and analyzed during the current study are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] M. Al-Shalabi, M. Anbar, and A. Obeidat, "Alternating sensing process to prolong the lifetime of wireless sensor networks," *Journal of Theoretical and Applied Information Technology*, vol. 97, no. 7, pp. 2132–2141, 2019.

[2] A. Obeidat and M. Al-shalabi, "An efficient approach towards network routing using genetic algorithm," *International Journal of Computers, Communications & Control*, vol. 17, no. 5, 2022.

[3] C. D. Xuan, H. Dinh, and T. Victor, "Malicious URL detection based on machine learning," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 1, 2020.

[4] Argus, "Argus and Machine Learning," 2022, http://www.qosient.com/argus.

[5] L. Böck, E. Vasilomanolakis, J. H. Wolf, and M. Muhlhauser, "Autonomously detecting sensors in fully distributed botnets," *Computers & Security*, vol. 83, pp. 1–13, 2019.

[6] K. Alissa, T. Alyas, K. Zafar, Q. Abbas, N. Tabassum, and S. Sakib, "Botnet Attack Detection in IoT Using Machine Learning," *Computational Intelligence and Neuroscience*, vol. 202214 pages, 2022.

[7] A. Kumar, M. Shridhar, S. Swaminathan, and T. J. Lim, "Machine learning-based early detection of IoT botnets using network-edge traffic," *Computers & Security*, vol. 117, Article ID 102693, 2022.

[8] M. Alauthman, N. Aslam, M. Al-kasassbeh, S. Khan, A. Al-Qerem, and K. K. Raymond Choo, "An efficient reinforcement learning-based Botnet detection approach," *Journal of Network and Computer Applications*, vol. 150, Article ID 102479, 2020.

[9] M. I. Kareem and M. N. Jasim, "Fast and accurate classifying model for denial-of-service attacks by using machine learning," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 3, pp. 1742–1751, 2022.

[10] T. F. Tu, J. Qin, H. Zhang, M. Chen, T. Xu, and Y. Huang, "A comprehensive study of Mozi botnet," *International Journal of Intelligent Systems*, vol. 37, no. 10, pp. 6877–6908, 2022.

[11] N. S. Akash, S. Rouf, S. Jahan, A. Chowdhury, and J. Uddin, "Botnet detection in IoT devices using random forest classifier with independent component analysis," *Journal of Information and Communication Technology*, vol. 21, no. 2, pp. 201–232, 2022.

[12] J. M. Ceron, K. Steding-Jessen, C. Hoepers, L. Granville, and C. Margi, "Improving iot botnet investigation using an adaptive network layer," *Sensors*, vol. 19, no. 3, p. 727, 2019.

[13] L. Duan, J. Zhou, Y. Wu, and W. Xu, "A novel and highly efficient botnet detection algorithm based on network traffic analysis of smart systems," *International Journal of Distributed Sensor Networks*, vol. 18, no. 3, Article ID 155014772110499, 2022.

[14] O. E. Taylor and P. S. Ezekiel, "A smart system for detecting behavioural botnet attacks using random forest classifier with principal component analysis," *European Journal of Artificial Intelligence and Machine Learning*, vol. 1, no. 2, pp. 11–16, 2022.

[15] S. I. Popoola, R. Ande, B. Adebisi, G. Gui, M. Hammoudeh, and O. Jogunola, "Federated deep learning for zero-day botnet attack detection in IoT-edge devices," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3930–3944, 2022.

[16] D. Stiawan, M. Y. Idris, R. F. Malik, S. Nurmaini, N. Alsharif, and R. Budiarto, "Investigating brute force attack patterns in IoT network," *Journal of Electrical and Computer Engineering*, vol. 2019, Article ID 4568368, 13 pages, 2019.

[17] M. C. Tran and Y. Nakamura, "Communication behaviour-based big data application to classify and detect HTTP automated software," *Journal of Electrical and Computer Engineering*, vol. 2016, pp. 1–11, 2016.

[18] Y. Li, Z. Li, and L. Chen, "Dynamic state estimation of generators under cyber attacks," *IEEE Access*, vol. 7, pp. 125253–125267, 2019.

[19] Y. Li, X. Wei, Y. Li, Z. Dong, and M. Shahidehpour, "Detection of false data injection attacks in smart grid: a secure federated deep learning approach," *IEEE Transactions on Smart Grid*, vol. 13, no. 6, pp. 4862–4872, 2022.

[20] Z. Qu, X. Bo, T. Yu et al., "Active and passivehybrid detection method for power CPS false datainjection attacks with improved AKF and GRU-CNN," *IET Renewable Power Generation*, vol. 16, no. 7, pp. 1490–1508, 2022.

[21] H. Qu, L. Lei, X. Tang, and P. Wang, "A lightweight intrusion detection method based on fuzzy clustering algorithm for wireless sensor networks," *Advances in Fuzzy Systems*, vol. 2018, Article ID 4071851, 12 pages, 2018.

[22] X. Liu, K. Li, W. Wang et al., "Improved RBF network intrusion detection model based on edge computing with multi-algorithm fusion," *International Journal of Computers, Communications & Control*, vol. 16, no. 4, 2021.

[23] A. Kumar Ramamoorthy and K. Karuppasamy, "Integration of fuzzy with incremental import vector machine for intrusion detection," *International Journal of Computers, Communications & Control*, vol. 17, no. 3, 2022.

[24] N. Kaur and M. Singh, "Botnet and botnet detection techniques in cyber realm," in *Proceedings of the 2016 International Conference on Inventive Computation Technologies (ICICT)*, IEEE, Coimbatore, India, August 2016.

[25] R. Limarunothai and M. A. Munlin, "Trends and challenges of botnet architectures and detection techniques," *Journal of Information Science and Technology*, vol. 5, no. 1, pp. 51–57, 2015.

[26] M. Antonakakis, T. April, M. Bailey et al., "Understanding the mirai botnet," in *Proceedings of the 26th USENIX security symposium (USENIX Security)*, Vancouver Canada, August 2017.

[27] M. Ramasamy and P. V. Eric, "An improved deep bagging convolutional neural network classifier for efficient intrusion detection system," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 1, pp. 405–413, 2022.