Optimized and Intelligent IDS for detecting the vulnerability of DDoS attacks in cloud environment

A Hariprasad Reddy[1], V.Shivanaraya Reddy[2], D Sudheer Reddy[3]

[1]Associate Professor,Dept.of.CSE, Geethanjali College of Engineering and Technology
[2]Associate Professor,Dept.of.CSE, Geethanjali College of Engineering and Technology
[3]Assistant Professor, Dept.of.CSE,Geethanjali College of Engineering and Technology

Received: 19.04.2024 Revised : 26.05.2024 Accepted: 29.05.2024

ABSTRACT
The proliferation of computing power and wide range of cloud services has resulted in an increase in the frequency and severity of cyberattacks. DDoS attacks, which target multiple hosts at a time, are gradually increasing due to the decentralized nature of cloud-based service delivery systems. In this study, a novel Intrusion Detection System (IDS) is presented for improving network performance by the accurate detection of Distributed Denial-of-Service (DDoS) attacks in cloud environment. The proposed IDS framework is equipped with a novel genetic algorithm based arithmetic optimization algorithm for attack vector selection and an intelligent ensemble Voting Classifier based on Woodpecker based Flamingo Search optimization algorithm for DDOS attack detection and classification. The proposed IDS outperforms various existing approaches in terms of network performance, DDOS attack prediction and mitigation rate.

Keywords: DDOS attacks, Cloud environment, Intrusion detection system, Optimization, Genetic algorithm

1. INTRODUCTION

The use of cloud computing as a service model has gained widespread acceptance in recent years. It makes it easy to access a variety of shared computing resources, such as storage, processing power, and applications, through the internet at any time and from anywhere in the globe. This accessibility is provided around the clock. These resources may be assigned to users on demand and made available to them in no time at all, and they only take a minimum amount of work to maintain [1]. It enables the enterprises to dynamically modify their capabilities in a cost-effective manner, which frees them from the burden of having to maintain an expensive information technology infrastructure. Focusing an organization's efforts on what it does best—its core business—can help that company achieve higher levels of productivity. In the most recent few years, cloud computing has seen a meteoric rise in popularity. A study report that was just released found that enterprises run 53% of their workload in the cloud, and that 50% of the data that organizations save is also stored on the cloud [2]. It is anticipated that by the year 2025, eighty percent of businesses would have moved their operations to the cloud [3]. The total amount spent on public cloud services is expected to reach $304.9 billion in 2021, which is an increase of 18.4% over the amount spent in 2020 ($257.5 billion). This estimate was made by Gartner [4]. Computing in the cloud has inspired the creation of a great number of apps that are geared at simplifying and improving many aspects of human existence. Some of these uses include online learning, online shopping, online entertainment, and online medical care [5-15]. It provides three different kinds of services: software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS), so that it can meet the requirements of a wide range of users (IaaS). This technology is more popular than conventional infrastructure-based systems [16-18] because to the benefits such as on-demand service, availability, scalability, and pay-as-you-go pricing. Despite this, a number of issues and limits need to be addressed, some of which include security and privacy, resource management, elasticity, response time, and many others [19, 25].

The most crucial problem that must be solved for this technology to be widely used is security. In addition to conventional security needs, the dynamic and scalable nature of the cloud presents certain unique security issues. Moreover, customers are not aware of the precise location of their data since it is controlled and kept remotely. Users must thus totally depend on the cloud provider to protect their data's security and privacy. Therefore, it becomes imperative for cloud providers to maintain security and privacy, since doing so might prevent the widespread use of cloud computing. Identity and access

management, digital signature, encryption and key management, attack detection and prevention, virtual environment security, interface security management, data storage security, hardware security, assurance and compliance measures, and attack detection and prevention are all ways that the cloud is made secure (as shown in Figure 1). This study, however, focuses on leveraging threat detection and prevention to secure cloud computing. DoS/DDoS attacks are significant security breaches that are used to interfere with cloud service availability. DoSattacks make about 22% of cloud attacks, claims the research [50]. DDoS attack instances are becoming more frequent every year. 4.83 million DDoS attacks are recorded during the first half of 2020, which is 15% greater than in 2019 [53]. By 2023, there will be 15.4 million attacks worldwide, predicts Cisco [54].
Therefore, it's essential to create effective defences against DDoS attacks. However, creating a DDoS attack detection system for the cloud is a difficult and complicated undertaking. There are a number of strategies for defending against these attacks in a cloud environment, but none of them can accurately identify these attacks, leaving room for the development of new DDoS attack detection techniques. In order to assure the availability of cloud services, this thesis investigates the many elements of DDoS attack detection for cloud computing. This research study tries to solve the problem of cloud computing security. In this sense, the effort focuses on creating defences against DDoS attacks. It also contains research done to suggest DDoS attack detection methods for cloud systems. This thesis presents machine learning-based defences against DDoS attacks. Only few forms of DDoS attacks are covered in this thesis. Additionally, this thesis exclusively uses simulation-based research to assess the security solutions that are suggested. This thesis only offers a few answers to the many problems in cloud security. The main contribution of this article includes the design and development of machine learning enthused IDS to mitigate the effect of DDoS attacks in cloud environment.

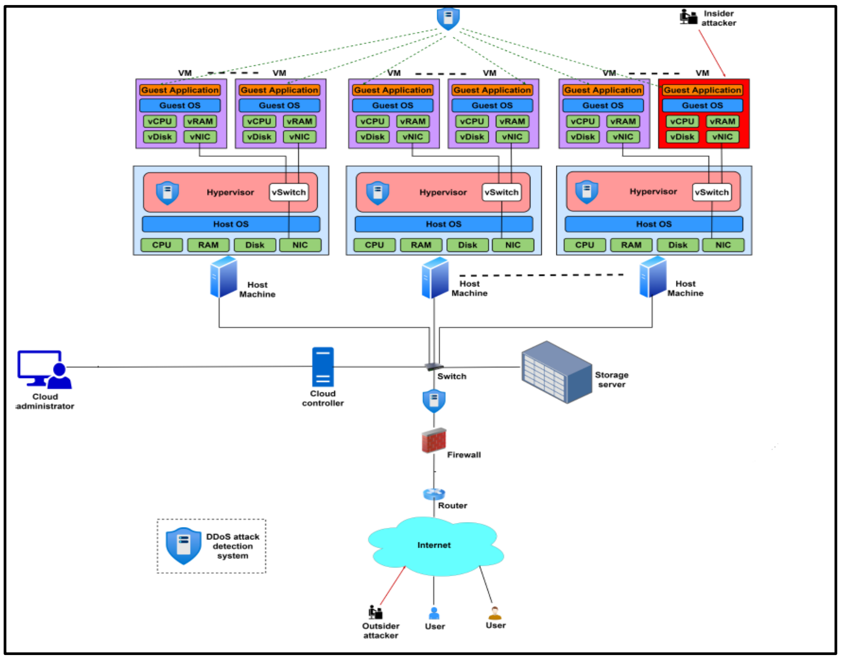


Figure 1: A Generic Cloud Security Architecture for DDoS attacks

2. Related Work

Signature-based intrusion detection systems identify ongoing suspicious activity by matching them to known harmful patterns or instructions. It keeps a record of the many different attacks and harmful behaviors in a signature database that it maintains. This signature database has to be updated on a regular basis in order to identify recently launched attacks. In order to identify potentially harmful behavior on a network, current network packets are compared with a previously saved set of rules. Snort, which was introduced by Martin Roesch, is both one of the simplest and most extensively used signature-based approaches (2015). Snort is a widely popular signature-based intrusion detection system (IDS) that can capture packets and monitor network traffic in real time. Figure 3 illustrates its primary components, which include a packet decoder, a pre-processor, a detection engine, a logging and warning system. The currently flowing traffic is pre-processed first, and then it is sent on to the detection engine. When preprocessing data, duplicate and incomplete information is removed. The current packet is then compared with records that are kept in the signature database by the detection engine. If there is a match, then the alert is produced for the relevant authority; otherwise, the packet is sent as a regular packet. If there is no match, then the alarm is not generated (Roesch, 2015).

A large number of studies provided signature-based IDS, such as the one that Lin et al. (2012) developed, in order to identify known threats in a cloud context. Information is gathered from the operating systems of all of the VMs and then dynamically updated so that the detection criteria may be configured. In their 2010 study, Lo et al. introduced a cooperative intrusion detection architecture (CGA). Every server has an intrusion detection system (IDS) that is a mix of a signature database and a block table that keeps a record of previous attacks. The packet is first compared to the block table, and then it is compared to known signatures. Recent attacks have a greater possibility of being present, thus it is important to examine them first. A threshold value is used to determine the severity of an abnormal packet via transfer alert clustering, which provides information about the aberrant packet. The intruder detection system (IDS) then discards the malicious packet. In a similar vein, Meng et al. (2014) suggested strategies for signature-based IDS. In a scenario including hostile network traffic, the authors state that the chance of a mismatch is much higher than the probability of a match happening. As a result, they identified the assault by using a policy called mismatch. In their 2015 study, Mandal and colleagues presented a signature-based IDS solution to identify application-level threats. This method involves inserting a sniffer between the user and the cloud provider. The sniffer then collects the data packets and sends them to the parser so that they may be further processed. The output of the parser is compared to the accumulated semantic rules by a parsing grammar, and the result is formed based on this comparison.

Despite the fact that a signature-based IDS can identify known threats quickly and has a very low incidence of false positives, it needs regular updating of its signature database. Anomaly detection methods were created in order to circumvent the constraints imposed by signature-based intrusion detection systems. This method creates a behavioral profile by analyzing the user's behavior in order to do so. After then, this behavioral profile is utilized to detect attacks, whether they are known or unknown. A fundamental example of the working concept for an anomaly detection method is shown in Figure 4. It is comprised of two primary parts, namely the phase of training, and the phase of detection. During the training phase, the feature creation module gathers input from the host system or network and preprocesses it so that features may be constructed. The training module will utilize these traits to construct a behavioral model of the subject. The data are classified according to this model as either normal or aberrant (intrusion) behavior. This model is what is used to identify intrusion during the anomaly detection phase. Any variation from the typical flow of traffic is regarded as an intrusion, and an alarm is triggered in the system that alerts the security administrator (Sari, 2015). This method can identify fresh threats, but it makes heavier demands on the computer's processing resources. Because every departure from typical behavior triggers an alert, it is now the responsibility of the security manager to determine the cause of the alarm. The classification of anomaly-based approaches is further broken down according to the method that is used to find anomalies. Some examples of such methods are machine learning, fuzzy logic, support vector machines, and data mining. In the most recent decade, a number of research have investigated the use of these procedures. To create their behavioral model, Kumar et al. (2011), for instance, made use of a hidden Markov approach. This method use a log file to monitor the frequency of system calls in order to identify potentially harmful activity. An IDS is built with the use of three different profiles, referred to as low, medium, and high, each of which correlates to different aspects of current activity. The low profile is used to depict patterns that have a little chance of being matched successfully. The high profile, on the other hand, refers to patterns that have a very high probability of being matched. In the end, the profile that best suits the present is the one that falls in the center. Every profile is matched using a threshold value that has been defined in advance.

Yuxin (2011) suggested a method that makes use of the machine-learning technology and is based on the static program behavior analysis. It operates in two stages: the first stage is to decode programs, and the second stage is to build context-free grammar in order to reflect the flow of the process. We investigate and piece together all of the branches so that we may acquire the whole sequences. The whole of the system calls have been condensed into a few concise sequences. They used two distinct methods for selecting features: one called Information Gain (IG), and the other was called Document Frequency (DF). An IDS method using a two-tier system was developed by Srinivasan et al. (2012). This method is a hybrid of unsupervised learning and supervised classification. For the purpose of intrusion detection, Wolthusen (2012) used the frequency of normal/abnormal system calls. To begin, throughout the course of some time period, a massive amount of records are gathered from each VM. The methos makes the assumption that the VMs are not going to be malicious for a certain amount of time once the startup process is complete. It has a temporal complexity of O (n), where n is the total number of lines in the expression. This method has a detection rate of one hundred percent, with just eleven percent being false positives.

3. Proposed Mechanism

IDSs are considered as an appropriate tool for monitoring and identifying data flow, and they provide protection against intrusions that compromise the availability, integrity, and confidentiality of an information system. The network intrusion detection technique for cloud computing environment that has been presented in this article is carried out based on the extraction of features and selection of those features using a voting based ensemble classifier. The CIC-DDOS 2019 dataset serves as the input for the analysis and has been pre-processed using encoding, scaling, and cleaning processes. After that, the dataset that has been normalized is balanced using the SMOTE approach, which involves oversampling. Using RNN (recurrent neural network inspired) t-Distributed Stochastic Neighbour embedding mechanism, the features are retrieved from the balanced dataset. The most advantageous features are selected by arithmetic optimization leveraged genetic algorithm (GAAOLGA). Further, the chosen feature sets are then divided into a training dataset and a testing dataset. These characteristics are categorized using the ensemble voting classifier technique that was suggested. Woodpecker and Flamingo Search Optimization (WPFS) integratedmetaheuristic algorithm is presented in this article to improve the accuracy of the classification.

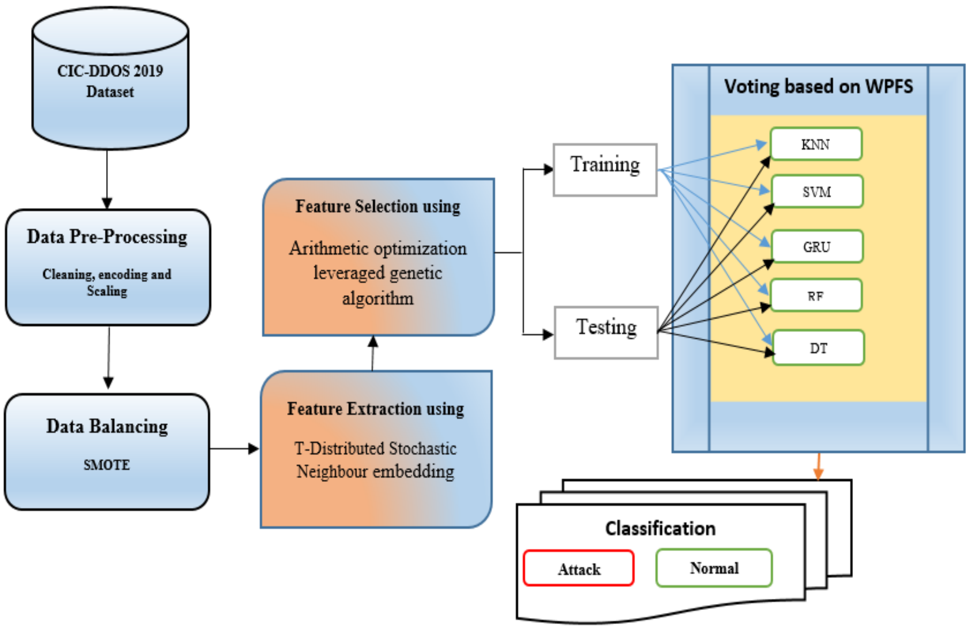


Figure 1: Block Diagram of Proposed Hybrid framework

3.1 Data Pre-Processing

As the dataset contains category values, numeric values, redundant and duplicate data, the proposed solution utilizes the pre-processing job. Processes like as data cleaning, data encoding, scaling, and normalization are performed on the raw data before the appropriate analysis can be made from them. The process of data cleaning involves removing any data that is redundant, irrelevant, incomplete, or erroneous before it is used for analysis. This helps the data become ready for analysis. The wrong format produces findings that are erroneous when using the typical data. The data cleansing topology improves accuracy without removing any information that is required. Because the computer can only read numerical data, the categorical values of the CIC-DDOS 2019 dataset are translated into numerical values. Consequently, all symbolic representations of characteristics are changed over to their corresponding numerical values throughout the process of data encoding. However, the performance of the classifier is negatively impacted because of the multiple scalings used for the features. It is vital to normalize the data since the characteristics might have very big numerical values. The characteristics are fixed using normalization to have a numerical value between 0 and 1, which might range from 0 to 1. In order to normalize the data, the Min-Max approach, which is the simplest and most straightforward option, is utilized in Eqn (1). A unified data format is available for the dataset after cleaning, encoding, and normalization have been completed.

$$\mathrm{K}:\mathrm{K}_{nor} = \frac{K - K_{\min}}{K_{\max} - K_{\min}} \qquad (1)$$

The dataset feature space (K) is signified by $U(\mathrm{K}_1, \mathrm{K}_2, ..., \mathrm{K}_n), 1 < i < I$. Where the total number of instances is denoted as I in the feature space.

3.2 Data Balancing

Because the class imbalance issue may have an effect on the pre-processed features, the SMOTE scheme is used in order to find a solution to the problem. Discovering an instance's k-nearest neighbours is one of the tasks that the SMOTE technique in [33]. After that, in order to generate a line segment in the feature space, it chooses at random an additional k-nearest neighbour to link with the first neighbours. In the process of finding the k-nearest neighbours, the Euclidean distance is used to sort the various cases. In the end, each instance is reset to the SMOTE class in order to produce synthetic instances. Synthetic instances are created primarily for the purpose of computing the difference between the original instance and its closest neighbour instance, multiplying that difference by a random integer ranging from 0 to 1, and then adding the result to the original instance. As a result, the issue of instances being over fit has been averted thanks to this. The considered dataset has a rather large number of normal occurrences when compared to the number of attack instances. If the dataset contains classes that are imbalanced, the accuracy of the classification will suffer as a result. As a result, the SMOTE approach is used to the dataset in order to bring it into better balance. Synthetic examples are produced in order to be added to the initial dataset, and then the classification model is trained using samples from the new dataset.

3.3. Feature Extraction using t-Distributed Stochastic Neighbour embedding mechanism (t-SNE)

This section introduces t-SNE algorithm for feature extraction based on hierarchical deep neural network. There are two stages involved in the t-SNE algorithm. In the first phase of the process, a probability distribution is carried out in a space with a high dimension. As a result, the ease with which a pair of items in the space may be picked is directly proportional to the degree to which they resemble one another. On the other hand, the likelihood of picking two different kinds of things is decreased. In addition, the probability in a low-dimensional space is generated, and the high-dimensional probability distribution is comparable to the low-dimensional probability distribution. When calculating similarity distances, many methods each employ their own unique set of criteria (such as k-means using Euclidean distance). In order to illustrate the similarity distances between two objects, the t-SNE technique relies on conditional probability. In this context, the pre-processed dataset ‘y’ which is given as an input for with N samples $y = (\mathrm{y}_{1,} \mathrm{y}_{2,} \mathrm{y}_{3,} ......\mathrm{y}_{\mathrm{n},})$ the distance between any two samples y_i and y_j will be determined based on Eqn.(2) & (3).

$$Q_{ij} = \frac{Q_{i|j} + Q_{\mathrm{j|i}}}{2N} \quad .... \ (2)$$

$$Q_{\mathrm{j|i}} = \frac{\exp(-\frac{\| y_i - y_j \|^2}{2\sigma^2{}_i})}{\sum_{k \neq 1} \exp(-\frac{\| y_i - y_k \|^2}{2\sigma^2{}_i})} \quad .... \ (3)$$

3.4 Feature Selection Using Genetic Algorithm Leveraged Arithmetic Optimization (GAAO)

When it comes to protecting the sensitive data in cloud computing networks, low-complexity and high-accuracy intrusion detection measures are crucial. As a result, the machine learning-based approach improves intrusion detection performance, and the detection rate is raised by the feature selection-based, nature-inspired optimization technique. The CIC-DDOS 2019 dataset has a large number of characteristics, some of which are useless, and a large number of features that were extracted from the dataset. The results show that the IDS classification performance is enhanced by the optimized feature selection approach. This is where the GAAO algorithm comes in to choose the best characteristics. Although the suggested feature selection approach ranks features according to their usefulness, it may not identify the most effective features for training a classifier. The suggested approach starts by sorting all characteristics into groups according to their usefulness in the classification processes. The features that were retrieved are split into a training set and a test set. The GAAO algorithm is used to analyse the training data and determine which characteristics are most useful. The classification algorithm is fed the best features extracted from the training and test data in order to evaluate the model's success. In order to minimize the error introduced by each iteration while simultaneously improving classification accuracy, Equation (4) models the ideal feature selection.

$$obj\_g(x) = \min E(y) \qquad (4)$$

$x = |y|,\ y \subseteq Y$ Where, denotes the collection of original characteristics (Y) chosen in order to achieve the best possible objective function.

The features are created in a subset at each GAAO population instance. For the fitness calculation, the chosen features are fed into classifiers like SVM kernel, KNN, Bi-LSTM, GRU, and RF. In such models, the optimum features are chosen by a minimization of the fitness function. Equation (5) defines error as the discrepancy between actual $h(k)$ and predicted output $\hat{h}(k)$.

$$N(k) = h(k) - \hat{h}(k), \qquad k = 1, 2, \ldots..n$$

Where K stands for experimental data. Equation (5), often known as the fitness function, calculates the fraction of the total errors by the total number of observations.

$$fitness(i) = \frac{\sum_{i=1}^{n} N(k)}{n} \qquad (5)$$

Best and worst fitness are used to determine the minimum and maximum fitness, respectively. By solving Equation, the local best value is determined during the exploratory search using equation 6. In its pursuit of the best answer, the GAAO algorithm constantly adjusts the parameters of Equation 7. In this step, the people who represent the minimal fitness solution are picked to continue into the following generation. This method gradually improves the classifier by including new characteristics. Once the best possible classification accuracy has been attained in the training dataset, the final choice is made on the optimal amount of features to use in the technique. Eventually, a feature set is settled upon for the proposed GAAO feature selection process.

$$y_{i,j}(y+1) = \begin{cases} a(y_j) \div (MOA + \in) \times \left(\left(u_j - l_j\right) \times \mu + l_j\right), & r_2 < 0.5 \\ a(y_j) \times MOA \times \left(\left(u_j - l_j\right) \times \mu + l_j\right), & otherwise \end{cases} \qquad (6)$$

Various sections of the search space have been arbitrarily explored using the Division (D) and Multiplication (M) techniques to find better answers. The model may be stated using the equation (5). During this time of searching, the MOA function is what determines the parameters. $r_1 < MOA$. The operator (D) is conditioned by $r_2 < 0.5$ and ignored by the operator (M) until the current task is completed. Where, $y_i(y+1)$ is represents the next iteration i^{th} solution, $y_{i,j}(y+1)$ represents the current iteration i^{th} solution in j^{th} position, and best obtained solution in j^{th} position is denoted as $b(x_j)$. Then the j^{th} position upper and lower bound value is denoted into u_j and l_j, respectively. The random numbers are denoted as r_1 and r_2, respectively. The search process is adjusted by the control parameter μ, the value is 0.5.

To carry out the exploitation searching phase, the operators S and A do a subtraction and an addition. Furthermore, with the additional stipulation that the present value of the asset is not greater than the MOA function, the exploitation will only occur $MOA(y)$. S and A operators perform in-depth search on various high-density locations in GAAOA, ultimately leading to the ideal solution defined by Equation (7)

$$y_{i,j}(y+1) = \begin{cases} a(y_j) - (MOP \times \left(\left(u_j - l_j\right) \times \mu + l_j\right), & r_3 < 0.5 \\ a(y_j) + MOP \times \left(\left(u_j - l_j\right) \times \mu + l_j\right), & otherwise \end{cases} \qquad (7)$$

Math Optimizer Probability (MOP) is a coefficient, and the value of the function at each iteration is denoted as $MOP(y)$, and the sensitive parameter is denoted as α.

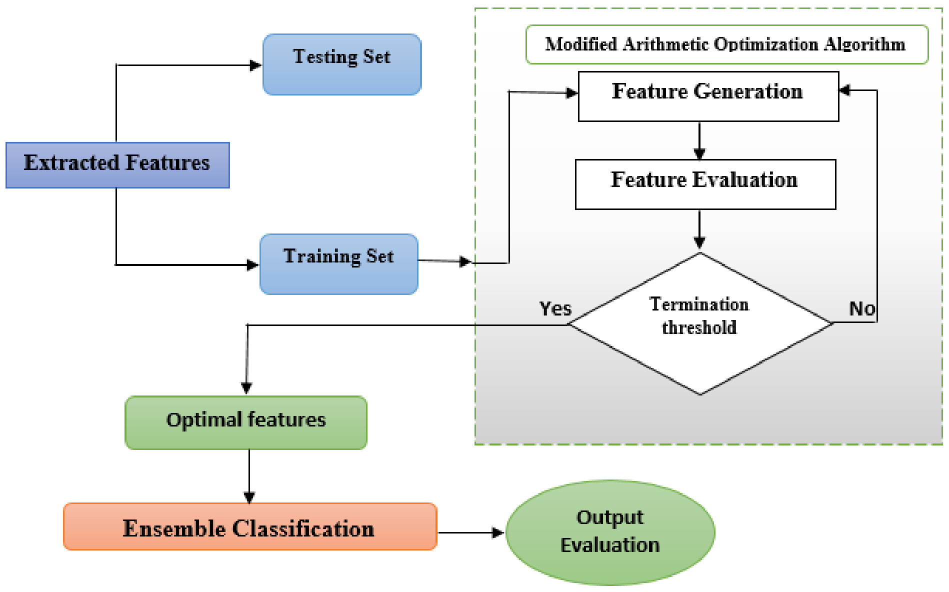


Figure 2: Proposed feature selection using Genetic algorithm leveraged Arithmetic Optimization

Pseudocode for proposed Genetic algorithm leveraged Arithmetic Optimization

```
Initialize: $\eta, \lambda$ and $j = 1, 2...., N$
While ($y <$ max(y)) do
Evaluate fitness and select best solution.
Compute $MOA(\mathrm{y}) =, \min + y \times \left( \frac{\max - \min}{\max \; y} \right)$      (8)
Compute $MOP(\mathrm{y}) = 1 - \frac{y^{1/\alpha}}{\max \; y^{1/\alpha}}$      (9)

for (i=1 to features) do
for (j=1 to locations) do
        Generate random values between (0,1)
if $r_2 > MOA$ then
Discovery stage
if $r_2 > 0.75$ then
update the positions in Eq. (6) based on the dividend operator D
else
update the positions in Eq. (6) based on the multiplication operator M
            end if
             Exploitation stage
            if $r_3 > 0.5$ then
update the positions in Eq. (6) based on the subtraction operator S
            else
update the positions in Eq. (6) based on the addition operator A
end if
end if
     end for
end for
$y = y + 1$
end while
Return best
```

3.5 Ensemble Voting Classifier based on Woodpecker based Flamingo Search optimization algorithm

For the purpose of network intrusion detection, which is used to examine the viability of the methods, ensemble learning offers a number of characteristics that make it superior than a single classifier. Following the feature selection process, the training data are then sent to the classifier to be trained. A voting classifier-based hybrid optimization technique is used in the suggested approach in order to categorize the network traffic that is being monitored. In order to improve the accuracy of the ensemble classifier, the Wood Peaker Optimization Algorithm (WPOA) is merged with the Flamingo Search Optimization (FSO). These ensemble machine learning classifiers use a combination of SVM kernel, KNN, Bi-LSTM, GRU, and RF. These four distinct machine learning classifiers, each of which is based on arithmetic optimization, are trained using the specified features. The voting method of the proposed hybrid WPOA with FSO algorithm, which is called WPFSO, optimizes the weight function of each classifier in order to achieve optimal performance (Wood Peaker based flamingo search optimization). The purpose of the ensemble classifier is to enhance classification accuracy while simultaneously decreasing error rates. As a consequence of the fact that individual classifiers are capable of producing varying outcomes, the ensemble classifier draws from the advantages offered by individual classifiers to achieve more accurate predictions.

In this hybrid method, an exploration technique is used to improve the overall best answer with each cycle of WPOA analysis. The goal of Hybrid WPFSO is to make the solution better by keeping the global best position up to date. If there is no improvement in the number of WPOA cycles, the previous best solution will be switched out for a new one that was calculated using the FSO technique. This new ideal level in fact transmits a signal to the math operator, which implies the math operator who was the previous best solution will be prompted to adjust its course in response to this signal. The advantages of both the WPOA's capacity to explore and the FSO's assets in terms of local search are included into the WPFSO algorithm. Although WPOA does not demonstrate improvement at the best point, FSO adds to the process of upgrading the best solution globally. In addition to that, this may take place several times throughout the process of finding a solution. The algorithm is kicked off with WPOA by creating a population and first setting the parameters for both methods. In WPOA, the FSO algorithm is used for the computation of the fitness solution throughout each iteration. If there is no improvement made to the current best solution throughout a cycle, then the FSO will use this best solution as the starting point for computing the best location. It will be carried on till the point when termination takes place. After then, the procedure for looking is carried out once more with WPOA using the now greatest solution in addition to the formerly finest solution. This procedure is repeated until the convergence requirements have been satisfied. The voting based WPFSO algorithm that has been suggested may be executed by following these steps.

```
Algorithm: Voting based WPFSO algorithm
Input : Various ML (Machine Learning ) MODELS ,  DDOS Optimal Attack Features
Output: Selection of best ML Model based on the Optimal Attack Features
p 1.    Initialize $\begin{bmatrix} V_{data} \leftarrow 0; \\ v_{index} \leftarrow 0; \\ T_a \leftarrow 0; \end{bmatrix}$ # voting data, voting index and total accuracy are initialized to '0';
p 2.    Initialize population of WPFSO based on the attack features
p 3.    Select randomized inputs from the attack features let us say if No. of feature F>1 then compute
        fitness function.
p 4.    If identified features < 3
p 5.      model←model.fit( train with all the parameters)
p 6.    Endif
p 7.    for model in MODELS do
p 8.    model←model.fit( train with all the parameters)
p 9.    model←model.predict (X_Test)
p 10.   accuracy ⇐metrics.accuracy_score(Y _Test, predict)
p 11.   VotingData ⇐VotingData + predict ∗accuracy
p 12.   TotAccuracy ⇐TotAccuracy + accuracy
p 13.   end for
p 14.   Check for improvement in the global optimal solution. If ( $P$ in iteration $i+1$ is not less than $P$
        in iteration $i$ ,
```

p 15. If the random number is not less than p *then move randomly* $F_i^{t+1} = \left(r^2 \times a_j^t - a_k^t\right) \times pF_i$

The present best solution is denoted as (b^*), *solution vector is denoted as* F_i^{t+1}, *iteration number, random number and fragrance is represented as* t, r *and* F_i, *respectively.* a_i^t *is used to update the butterfly position during iterations.*

p 16. The global optimal solution is obtained then go to step 3 and repeat steps until the termination criteria will met.

In the WPFSO method that has been presented, the best possible outcome is always given to the output after each execution and prognosis of a new instance. In conclusion, the output that is ideal is accomplished while maintaining the greatest possible accuracy and experiencing the fewest possible mistakes.

4. RESULTS AND DISCUSSION

In order to validate the proposed methodology, the CIC-DDOS 2019 dataset was used. Experiments are done on a computer with a 2.3 GHz Intel Core i3 CPU, and the MATLAB R2020a tool is used for both implementation and assessment of the experiments. The synthetic dataset used in CIC-DDOS 2019 [13] was generated with the help of the freeware application NetSim. This tool is capable of CC, LTE networks, Cognitive radio networks, and a variety of other network types. The gateway, wired nodes, sensor nodes, and routers are the components that make up the CC network. Following the engagement of the captured packets in their respective CSV distinct files for each assault, the aforementioned files are then merged into the CIC-DDOS 2019 dataset. It has twenty different numbers of features and two label qualities in addition to those. Table 1 contains a listing of the dataset's characteristics, while the dataset itself contains seven different types of attacks, including Sybil, Clone ID, Hello flooding, Local repair, Selective forwarding, Sinkhole, and Blackhole.

Table 1. Feature set utilized from the dataset

Id number	Feature name
3	Control Packet Type
4	Source ID
5	Destination ID
6	Transmitter ID
7	Receiver ID
8	APP Layer Payload
9	TRX Layer Payload
10	NW Layer Payload
11	MAC Layer Payload
12	PHY Layer Payload
13	PHY Layer Overhead
14	APP Layer Arrival Time
15	TRX Layer Arrival Time
16	NW Layer Arrival Time
17	MAC Layer Arrival Time
18	PHY Layer Arrival Time
19	PHY Layer Start Time
20	PHY Layer End Time

The dataset is partitioned into categories in accordance with the breakdown shown in Table 2. As is customary, the dataset is partitioned, and successful attacks are counted toward testing and training persistence. There have been 42,587 total incidents of attacks.

Table 2. Category wise dataset description

Category	Counts
Clone_ID	5852
Hello_fooding	5854
Local_repair	5858

Selective_forwarding	5625
Sinkhole	5898
Blackhole	5454
Sybil	5255
Normal	53696

Table 3. Testing and Training dataset description

Label	Training dataset	Testing dataset
Attack	42227	17581
Normal	125689	54656

4.1 Performance Metrics

The performance of the proposed NIDS was evaluated using a few of the matrices that are discussed further down. The goal of this study based on NIDS is to achieve the highest possible level of accuracy in predicting occurrences from the test dataset. Accuracy, True positive (TP), False positive (FP), True Negative (TN), False Negative (FN), Recall, Precision, and F1 score are the metrics that are used to evaluate the performance. Equation 10 [36] is what is used to determine how accurate something is.

$$A = \frac{TN + TP}{FP + FN + TP + TN} \qquad (10)$$

In this scenario, TP is defined as the percentage of attacks that are properly recognized, FP as the percentage of attacks that are mistakenly identified, TN as the percentage of attacks that are correctly identified, and FN as the percentage of attacks that are inaccurately identified.

$$precision(p) = \frac{TP}{TP + FP} \qquad (11)$$

$$recall(r) = \frac{TP}{TP + FN} \qquad (12)$$

$$Fscore = 2\frac{p \times r}{p + r} \quad (13)$$

4.2 Balancing Data Using SMOTE

There are fewer attack classes available than there are standard class slots available. As a result, the dataset has to be balanced, hence the oversampling approach of SMOTE is applied. In the SMOTE method, the closest neighbor parameter value k=4 is used to apply an oversampling technique to the data in order to normalize it and make it comparable to other classes. Because of the performance of SMOTE, the number of attack instances has been raised to 142586 features. The performance of the SMOTE contributes to an improvement in the accuracy of the NIDS system. In the absence of SMOTE balance, the precision is subpar, and the margin of error is significant. Table 4 presents the results of a comparison between the performance with and without balancing. Based on Table 4, the suggested approach has obtained an accuracy of 98% while balancing, whereas the accuracy drops to 95% when balancing is not performed.

Table 4. Comparison of SMOTE balancing and without balancing performance

Parameters	With balancing	Without balancing
Accuracy	0.9849	0.9564
Error	0.0215	0.0482
Sensitivity	0.9709	0.9283
Specificity	0.9867	0.996
Precision	0.9747	0.9284
FPR	0.1057	0.0986
F1_score	0.9723	0.9271
MCC	0.9701	0.9216
Kappa	0.9193	0.7926

Features are extracted from the dataset using the t-Distributed Stochastic Neighbour embedding mechanism method after the dataset has been balanced. There are just 14 characteristics to work with before extraction, but thereafter there are 365. The feature selection technique chooses the most useful characteristics from among those that have been retrieved.

4.3 Feature Selection Using Genetic algorithm leveraged Arithmetic Optimization

The GAAO algorithm is used in order to choose the features in the most effective way possible. This allows for improved feature selection. The fitness function is used to choose the best features, which implies that a low error rate suggests the best feature set from the dataset. Figure 3 displays the convergence curve of the suggested method, which demonstrates that AOA has a superior ability for exploration. The GAAO method has reached the point where the minimal average error value of 0.0183 has been established in order to choose the appropriate features. When contrasted with GA, PSO, SHO, and CSA, as well as ALO, the feature selection algorithms with the least amount of error for the present situation are listed in descending order from best to worst. The accuracy of feature selection has been improved because to the GAAO approach that was presented. The algorithm's parameters are detailed in Table 5.

Table 5. GAAO Parameters

GAAO	
Parameters	**Values**
Search solutions	36
Maximum iteration	125
MOP max	1
MOP	0.23
μ	0.633
Lower and upper bound	0 and 1

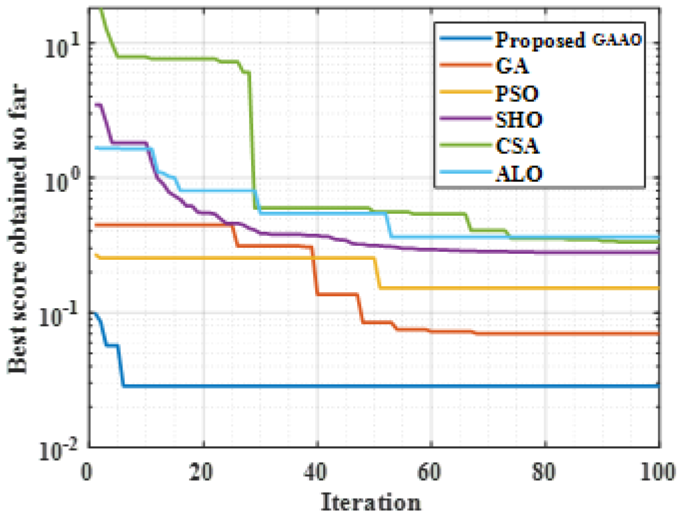


Figure 3. Convergence Comparison

4.4 Performance analysis of Voting Based Classifier

An ensemble classifier that makes use of the SVM kernel, kNN, RF, Bi-LSTM, and GRU was suggested for use with IDS. The multiple classifier model has included a voting system based on majority rule in order to pick the optimal course of action. When it came time to make the final forecast, a weighted majority vote was used to aggregate all of the previous guesses about how the classification would turn out. A weighted version of the data was used to develop a training sequence that was then delivered in a sequential fashion to basic learners. As the basis classifier, the k-nearest neighbours' algorithm is used. The WPFSO algorithm's hybrid method was used to perfect the optimization of the weight function. Instances of normal attacks within the dataset are given the weights that have been optimized for them. Normal attacks in the dataset are then given the optimum weights. Performance of the proposed WPFSO algorithm is shown in Figure 4. Comparisons to other algorithms including GA, PSO, SHO, CSA, and ALO have shown the WPFSO algorithm's efficacy. The Mean Squared Error (MSE) is a metric used to measure how well a classifier performs by comparing the classifier's actual output to the desired output. You may write it down as an equation (14),

$$MSE = \sum_{i=1}^{n} (o_i^t d_i^t)^2 \qquad (14)$$

Table 6. Performance analysis of voting based proposed WPFSO algorithm

Parameters	**Proposed**	**GA**	**PSO**	**SHO**	**CSA**	**ALO**
Accuracy	0.9957	0.9761	0.9518	0.8992	0.8811	0.83
Sensitivity	0.9937	0.9754	0.9497	0.9109	0.8911	0.8305
Specificity	0.9996	0.9968	0.9933	0.9858	0.9832	0.9759
Precision	0.989	0.9694	0.9453	0.8926	0.8744	0.8232
F1_score	0.9889	0.9694	0.9447	0.8951	0.8767	0.8231
MCC	0.9881	0.9662	0.9379	0.8833	0.8619	0.7995
Kappa	0.9498	0.8601	0.749	0.5085	0.4259	0.1922
FPR	0.0016	0.0044	0.0079	0.0154	0.018	0.0253
Error	0.0115	0.0311	0.0554	0.108	0.1261	0.1772

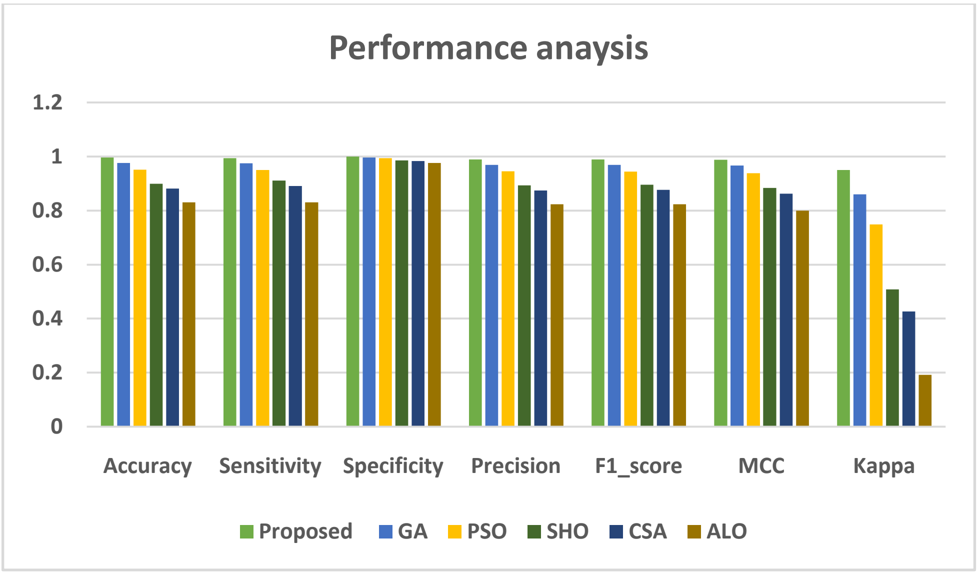


Figure 4. Performance analysis

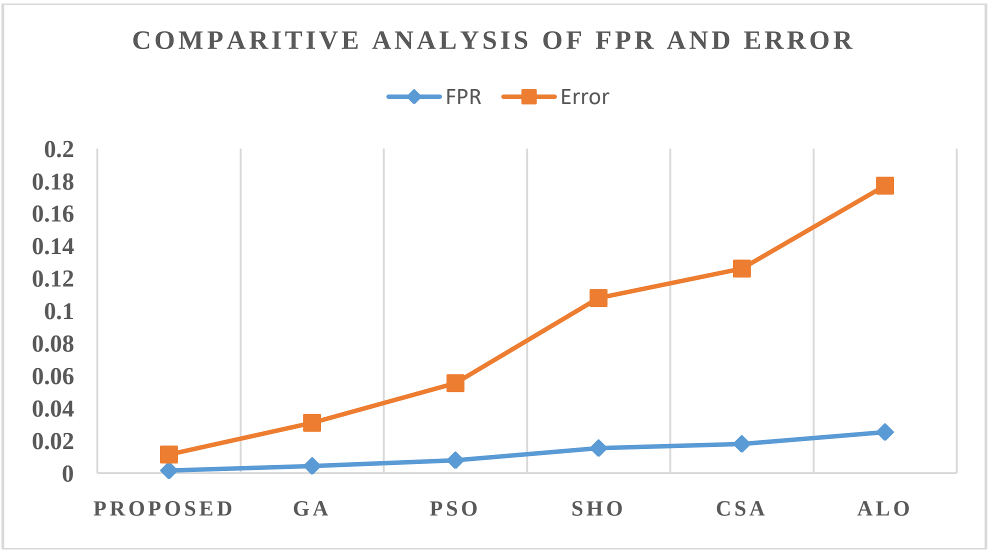
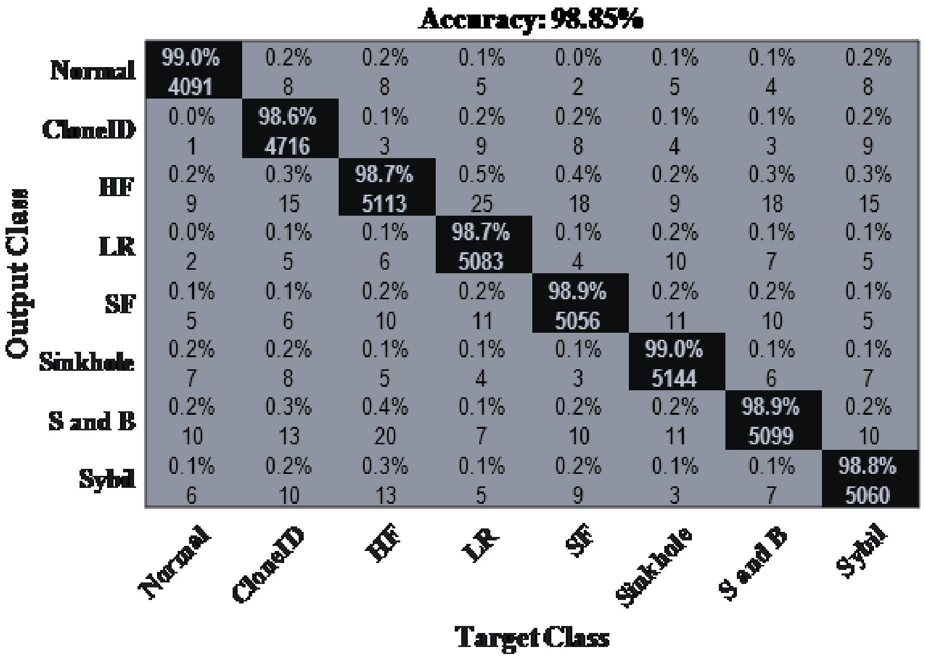


Figure 5. FPR VS Error Rate

The effectiveness of the proposed classifier is measured by its accuracy relative to those obtained using the individual methods (RF, kNN, SVM kernel, GRU, and Bi-LSTM based ensemble classifier). The A-BO weighted majority voting method outperformed other approaches on the dataset. A confusion matrix depicts the voting classifier's precision. Existing voting algorithms such as GA, PSO, SHO, CSA, and ALO are compared to the proposed technique, demonstrating its superior accuracy. GA, PSO, SHO, CSA, and ALO voting accuracy is 96.79, 94.50, 89.45, 88.53, and 82.57 percent, respectively. The suggested voting algorithm, however, has attained an accuracy of 98.17 percent, putting it ahead of the other methods. The confusion matrix for both new and current voting classifiers is shown in Figure 6.


Accuracy: 98.85%

Output Class / Target Class

	Normal	CloneID	HF	LR	SF	Sinkhole	S and B	Sybil
Normal	99.0% 4091	0.2% 8	0.2% 8	0.1% 5	0.0% 2	0.1% 5	0.1% 4	0.2% 8
CloneID	0.0% 1	98.6% 4716	0.1% 3	0.2% 9	0.2% 8	0.1% 4	0.1% 3	0.2% 9
HF	0.2% 9	0.3% 15	98.7% 5113	0.5% 25	0.4% 18	0.2% 9	0.3% 18	0.3% 15
LR	0.0% 2	0.1% 5	0.1% 6	98.7% 5083	0.1% 4	0.2% 10	0.1% 7	0.1% 5
SF	0.1% 5	0.1% 6	0.2% 10	0.2% 11	98.9% 5056	0.2% 11	0.2% 10	0.1% 5
Sinkhole	0.2% 7	0.2% 8	0.1% 5	0.1% 4	0.1% 3	99.0% 5144	0.1% 6	0.1% 7
S and B	0.2% 10	0.3% 13	0.4% 20	0.1% 7	0.2% 10	0.2% 11	98.9% 5099	0.2% 10
Sybil	0.1% 6	0.2% 10	0.3% 13	0.1% 5	0.2% 9	0.1% 3	0.1% 7	98.8% 5060

Figure 6. Confusion Matrix depicting the Performance of the proposed WPFSO Algorithm

The effectiveness of the proposed WPFSO voting classifier scheme is also compared with other ensemble techniques. Adaptive boosting and bagging techniques are compared with WPFSO voting classifier to show the efficiency of the proposed algorithm. The error of the proposed method (0.0321) is lower than the existing methods Ada boost (0.912) and bagging (0.078). Also, the performance of sensitivity, specificity, precision, FPR, F1_score, MCC, and kappa is outperformed the proposed method compared to existing approaches. Table 7 displays the comparison of the proposed and existing voting classifier.

Table 7. Comparison of the proposed and existing voting classifier

Parameters	Voting WPFSO	Ada boost	Bagging
Accuracy	0.984	0.9473	0.9243
Sensitivity	0.953	0.9176	0.891
Specificity	0.9967	0.9936	0.9905
Precision	0.9596	0.9156	0.8835
F1_score	0.9564	0.9159	0.8856
MCC	0.9483	0.9052	0.8722
Kappa	0.8768	0.772	0.6671
Error	0.0321	0.055	0.078
FPR	0.0045	0.0076	0.0107

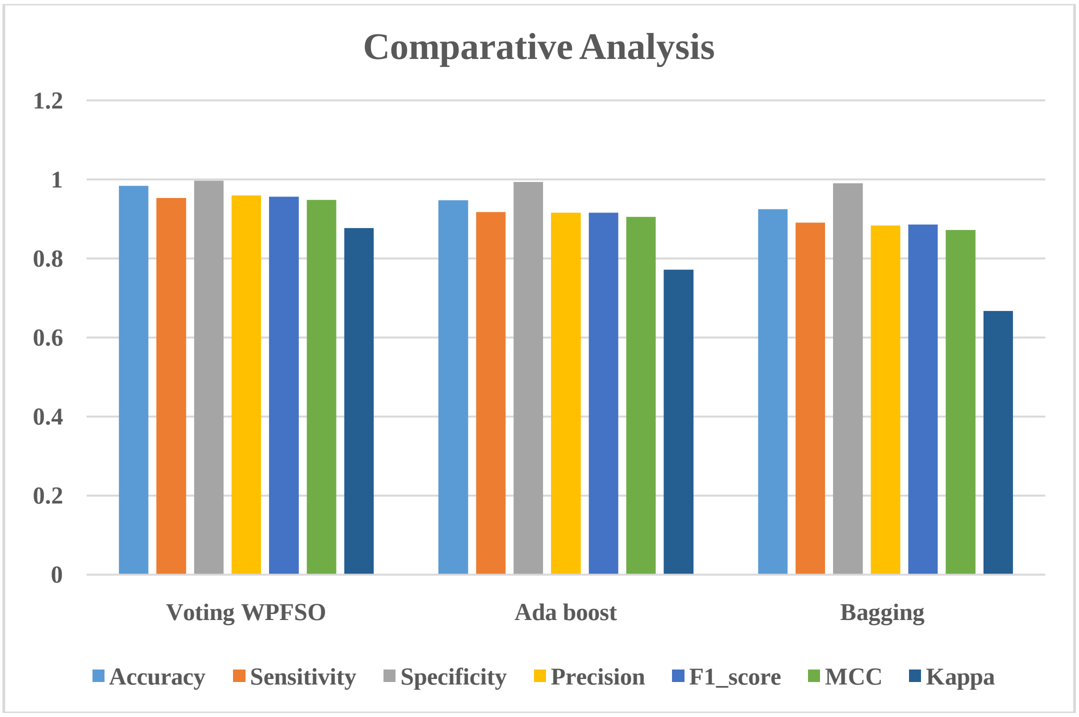


Figure 7. Comparative analysis of the proposed and existing voting classifier

5. CONCLUSION

This paper presents a novel Intrusion Detection System (IDS) is presented for improving network performance by the accurate detection of Distributed Denial-of-Service (DDoS) attacks in cloud environment. The proposed IDS framework is equipped with a novel genetic algorithm based arithmetic optimization algorithm for attack vector selection and an intelligent ensemble Voting Classifier based on Woodpecker based Flamingo Search optimization algorithm for DDOS attack detection and classification. The proposed IDS outperforms various existing approaches in terms of network performance, DDOS attack prediction and mitigation rate. The accuracy rate of attack detection in proposed system is 98.4% which is comparably best than existing approaches.

REFERENCES

[1] J. He, “Cloud computing load balancing mechanism taking into account load balancing ant colony optimization algorithm,” Computational Intelligence and Neuroscience, vol. 2022, Article ID 3120883, 10 pages, 2022.
[2] F. S. Al-Anzi, S. K. Yadav, and J. Soni, “Cloud computing: security model comprising governance, risk management and compliance,” in Proceedings of the 2014 International Conference on Data Mining and Intelligent Computing (ICDMIC), September 2014.
[3] N. Mishra, R. K. Singh, and S. K. Yadav, “Design a new protocol for vulnerability detection in cloud computing security improvement,” in Proceedings of the International Conference on Innovative Computing & Communication (ICICC) 2021, Delhi, India, February 2021.
[4] D. Alsmadi and V. Prybutok, “Sharing and storage behavior via cloud computing: security and privacy in research and practice,” Computers in Human Behavior, vol. 85, pp. 218– 226, 2018.
[5] J. Shroff, R. Walambe, S. K. Singh, and K. Kotecha, “Enhanced security against volumetric DDoS attacks using adversarial machine learning,” Wireless Communications and Mobile Computing, vol. 2022, Article ID 5757164, 10 pages, 2022.
[6] O. Cepheli, S. B¨uy¨ukçorak, and G. Karabulut Kurt, “Hybrid ¨ intrusion detection system for ddos attacks,” Journal of Electrical and Computer Engineering, pp. 1–8, 2016.
[7] B. A. Khalaf, S. A. Mostafa, A. Mustapha et al., “An adaptive protection of flooding attacks model for complex network environments,” Security and Communication Networks, vol. 2021, Article ID 5542919, 17 pages, 2021.
[8] M. MyintOo, S. Kamolphiwong, T. Kamolphiwong, and S. Vasupongayya, “Advanced support vector machine- (ASVM-) based detection for distributed denial of service (DDoS) attack on software defined networking (SDN),” Journal of Computer Networks and Communications, Article ID 8012568, 12 pages, 2019.

[9] X. Yu, W. Yu, S. Li, X. Yang, Y. Chen, and H. Lu, "WEB DDoS attack detection method based on semisupervised learning," Security and Communication Networks, vol. 2021, Article ID 9534016, 10 pages, 2021.

[10] U. A. Butt, M. Mehmood, S. B. H. Shah et al., "A review of machine learning algorithms for cloud computing security," Electronics, vol. 9, no. 9, p. 1379, 2020.

[11] M. Idhammad, K. Afdel, and M. Belouch, "Detection system of HTTP DDoS attacks in a cloud environment based on information theoretic entropy and random forest," Security and Communication Networks, vol. 2018, Article ID 1263123, 13 pages, 2018.

[12] M. Eshtay, H. Faris, and N. Obeid, "A competitive swarm optimizer with hybrid encoding for simultaneously optimizing the weights and structure of Extreme Learning Machines for classification problems," International Journal of Machine Learning and Cybernetics, vol. 11, no. 8, pp. 1801– 1823, 2020.

[13] N. Mishra and R. K. Singh, "DDoS vulnerabilities analysis and mitigation model in cloud computing," Journal of Discrete Mathematical Sciences and Cryptography, vol. 23, no. 2, pp. 535–545, 2020.

[14] C. Amine, S. Ben Alla, and A. Ezzati, "Makespan optimisation in cloudlet scheduling with improved DQN algorithm in cloud computing," Scientific Programming, vol. 2021, Article ID 7216795, 11 pages, 2021.

[15] A. Amjad, T. Alyas, U. Farooq, and M. Tariq, "Detection and mitigation of DDoS attack in cloud computing using machine learning algorithm," ICST Transactions on Scalable Information Systems, vol. 0, no. 0, Article ID 159834, 2018.

[16] A. Aliyu, A. H. Abdullah, O. Kaiwartya et al., "Mobile cloud computing: taxonomy and challenges," Journal of Computer Networks and Communications, Article ID 2547921, 23 pages, 2020.

[17] P. Singh and V. Ranga, "Attack and intrusion detection in cloud computing using an ensemble learning approach," International Journal of Information Technology, vol. 13, no. 2, pp. 565–571, 2021.

[18] H. A. Mahmood and S. H. Hashem, "Network intrusion detection system (NIDS) in cloud environment based on hidden na¨ıvebayes multiclass classifier," Al-Mustansiriyah Journal of Science, vol. 28, no. 2, pp. 134–142, 2018.

[19] Y. Jiang, B. Qiu, C. Xu, and C. Li, "&e research of clinical decision support system based on three-layer knowledge base model," Journal of healthcare engineering, Article ID 6535286, 8 pages, 2017. [20] Y. Berguig, I. Laassiri, and S. Hanaoui, "DoSdetection based on mobile agent and na¨ıvebayes filter," in Proceedings of the 2018 International Symposium on Advanced Electrical and Communication Technologies (ISAECT), Rabat, Morocco, November 2018.

[20] S. Nandi, S. Phadikar, and K. Majumder, "Detection of DDoS attack and classification using a hybrid approach," in Proceedings of the 2020 >ird ISEA Conference on Security and Privacy (ISEA-ISAP) 2020, pp. 41–47, Guwahati, India, March 2020.

[21] J. Kim, J. Kim, H. Kim, M. Shim, and E. Choi, "CNN-based network intrusion detection against denial-of-service attacks," Electronics, vol. 9, no. 6, p. 916, 2020.

[22] A. E. Cil, K. Yildiz, and A. Buldu, "Detection of DDoS attacks with feed-forward based deep neural network model," Expert Systems with Applications, vol. 169, no. December 2020, Article ID 114520, 2021.

[23] A. Rangapur, T. Kanakam, and A. Jubilson, "DDoSDet: an approach to Detect DDoS attacks using Neural Networks," 2022, https://arxiv.org/abs/2201.09514.

[24] A. Saroha and A. Singh, "Security concerns and countermeasures in cloud computing: a qualitative analysis," International Journal of Information Technology, vol. 11.4, pp. 683–690, 2019.

[25] R. Goel, M. Garuba, and A. Girma, "Cloud computing vulnerability: DDoS as its main security threat, and analysis of IDS as a solution model," in Proceedings of the 2014 11th International Conference on Information Technology: New Generations, Las Vegas, NV, USA, April 2014.

[26] R. V. Deshmukh and K. K. Devadkar, "Understanding DDoS attack & its effect in cloud environment," Procedia Computer Science, vol. 49, pp. 202–210, 2015.

[27] M. Masdari and M. Jalali, "A survey and taxonomy of dos attacks in cloud computing," Security and Communication Networks, vol. 9, 2016.

[28] P. Oberoi, "Survey of various security attacks in clouds based environments," International Journal of Advanced Research in Computer Science, vol. 8, no. 9, pp. 405–410, 2017.

[29] E. S. JeyaJothi, J. Anitha, S. Rani, and B. Tiwari, "A comprehensive review: computational models for obstructive sleep apnea detection in biomedical applications," BioMed Research International, pp. 1–21, 2022.

[30] N. O. Ogwara, K. Petrova, and M. L. Yang, "Towards the development of a cloud computing intrusion detection framework using an ensemble hybrid feature selection approach," Journal of Computer Networks and Communications, vol. 2022, Article ID 5988567, 16 pages, 2022.

[31] UNB, "NSL-KDD dataset," https://www.unb.ca/cic/datasets/ nsl.html.

[32] Kaggle, "KDD-CUP-98 Dataset," https://www.kaggle.com/ datasets/towhidultonmoy/kddcup98-dataset.

[33] Kaggle, "NSL-KDD randomforest w/optuna,".

[34] L. Dhanabal and S. P. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," International journal of advanced research in computer and communication engineering, vol. 4.6, pp. 446– 452, 2015.

[35] M. Li and D. Han, X. Yin, H. Liu, and D. Li, Design and implementation of an anomaly network traffic detection model integrating temporal and spatial features," Security and Communication Networks, vol. 2021, Article ID 7045823, 15 pages, 2021.

[36] S. K. Yadav, D. K. Tayal, S. N. Shivhare, and S. NareshShivhare, "Perplexed Bayes classifier-based secure and intelligent approach for aspect level sentiment analysis," International Journal of Advanced Intelligence Paradigms, vol. 13, no. 1/2, p. 15, 2019.

[37] A. UlHaq, J. P. Li, M. H. Memon, S. Nazir, and R. Sun, "A hybrid intelligent system framework for the prediction of heart disease using machine learning algorithms," Mobile Information Systems, vol. 2018, Article ID 3860146, 21 pages, 2018.

[38] C. S. Carlos, "Perplexed Bayes classifier," in Proceedings of the 12th International Conference on Natural Language Processing, Trivandrum, India, December 2015.

[39] K. Dhingra and S. K. Yadav, "Spam analysis of big reviews dataset using fuzzy ranking evaluation algorithm and hadoop," International Journal of Machine Learning and Cybernetics, vol. 10.8, pp. 2143–2162, 2019