ISSN 2007-9737

DDoS Attacks in Traffic Flow Streams Using Ensemble Classifiers

Dutta Sai Eswari*, Panga V. Lakshmi

GITAM University, Department of CSE,
India

saieswari3@gmail.com,vpanga@gitam.edu

Abstract. The failure of internet networking systems, which can happen in various methods, may negatively impact contemporary information and communication technologies. In these circumstances, DDoS attacks have targeted a growing number of organizations. These attacks use a deluge of demands for computation and communication resources to order a service unavailable to genuine users. Distributed denial-of-service (DDoS) attacks must be prevented on vital resources. This manuscript's most important new development is the DDoS attack defence ensemble classifier model. The suggested model uses these specifications to enable a drift detection feature and includes defining streaming properties for service requests. Additionally, an ensemble classifier is used to detect changes in the pattern of service request traffic. Statistical metrics like true negative rate, positive predictive value, and accuracy were used to test and analyze the service request stream synthesis results. Comparing the model to other benchmark models discussed in recent academic works would have been another tactic that could have been used to increase the model's importance.

Keywords. Distributed denial of service (DDoS) attacks, ensemble classifier, drift detection.

1 Introduction

The way news spreads has significantly changed as a result of developments in Technology (ICT). The way people receive and share information has significantly changed due to the information flow through the internet. The services supported by ICT are improving information exchange and continuum operations. Information and communication technology (ICT) system issues have a negative effect on many facets of contemporary society and the digital world.

The failure of these networks can be attributed to several things, including viruses and unauthorized users. DDoS attacks, or distributed denial-of-service attacks, involve the coordinated use of numerous computers and networks to deny access to authorized users. Over the previous ten years, it has constantly been a threat.

Therefore, these evil attempts to seize vital resources must be stopped. These DDoS assaults demand an ensemble-based classifier with minimal computational requirements and high decision-making accuracy in ambiguous data. The scholar suggests a method known as distributed denial of service attacks to model the distribution statistics of network flows, identify unusual attacks, and meet the network's requirements.

Numerous new developments and considerable challenges exist as the system network expands exponentially and the internet networks become more complex. The behaviour of network attacks significantly increases the difficulty of discovery. IT security experts are growing concerned about distributed request floods, which are used to deny service to one or more target servers. Numerous reasons could be responsible for such floods.

Distributed denial-of-service (DDoS) distribution, which has a global scope, can mitigate some elements of distributed denial-of-service (DDoS) attacks but not all of them. For example, telco clouds and networks need to be protected at a more granular level, but this is not feasible. This is because specific characteristics call for targeted, community-based intervention.

Signature-based detection and mitigation techniques are frequently used to thwart DDoS assaults. This is accurate to a very large degree. Their independence in understanding the distinctions between regular and attack traffic patterns is crucial to their success on the battlefield. Despite their inventiveness in defending against typical attack types, this is where they excel. The accuracy of conventional detection models for detecting DDoS patterns has dropped due to the emergence of new security threats.

Targeted zombie armies that are remotely controlled can help mitigate distributed denial of service (DDoS) assaults. When cybercriminals initiate an attack of this nature, they fully exploit the compromised system by stealing its data or utilizing all of its memory and bandwidth. Because distributed denial of service assaults only utilize a small portion of the available computational resources, they have been proven ineffective and expensive [1].

DDoS attacks, also known as distributed denial of service attacks, are increasing across all kinds of networks, including those used for cloud computing, according to recently gathered data. On modern networks, distributed denial of service attacks [2] appear in variousflavours [3]. RUDY and control packet overflow are only two potential attack types. (R-U-Dead-Yet) With RUDY, attacking a target server is as easy as flooding it with many infinitely lengthy sessions.

When this occurs, there is no longer any responsiveness to fresh queries from other users. A more recent attack called HTTP POST/GET [4] sends legally posted messages to a web server that hosts an application very slowly. The web service will experience prolonged downtime and possibly even failure if subjected to a distributed denial of service attack.

This kind of assault, which can come from numerous other vectors, is modelled after the SQL injection attack paradigm (SIDDoS). The model [5] then tries to access database resources the opponent cannot access. This research discovered that the mean square distance for achieved features decreased as the number of sessions in the network traffic corpus increased. Given the exponential increase in the network traffic corpus, this outcome shouldn't be unexpected.

Because the data are probabilistic rather than deterministic, features determined to be insignificant are a consequence of this.

The rest of the article is prearranged as follows: section 2 discusses the related work, section 3 proposes a defence of DDoS attack in traffic flow streams, section 4 deliberates the experimental outcomes, and section 5 concludes the research paper.

2 Related Work

There are two main categories for DDoS intrusion monitoring systems: analysis of the data source being transferred and traffic analysis. The most common detection and mitigation models and their benefits and drawbacks will be discussed below.

2.1 Analysis of the Transmitted Data Source

The preliminary investigation of IDS is the main topic of this part of [6], which also chronologically lists important DoS. The attacks SYN-Flood, HTTP-flood, and Smurf are just a few that have significantly altered the timeline. Most of this study falls into one of three categories: anomaly-based tactics, signature-based techniques, or a combination of both [7].

Neural networks have been created and usedto spot possible real-time distributed denial of service attacks. Researchers discovered layer seven DDoS flooding attacks were extremely potent in frequency and quantum in the study mentioned in [8].

These models' intrusion detection accuracy rates, however, are generally insufficient. Only the NB classifier can learn and build with a significant precision rate while operating at considerably faster speeds than other benchmark approaches [9, 10] studied the dynamic interaction between malicious traffic patterns' temporal and geographic correlation.

This model detected malicious traffic flows without needing router-level adjustments to IP forwarding schemes. Researchers in [11] used ensemble NNs composed of RBF and SOMs to identify malicious traffic and achieved remarkably high accuracy rates.

The success of the research was largely due to the use of an ensemble of NNs. The referenced research [12] claims that the accuracy of the multilayer perception NN was dependent on neurons present in the omitted layers. Results from the study's trials revealed accuracy rates above 90%, which is regarded as sufficient in the industry. The authors of [13] employed a bi-layer feed-forward neural network to distinguish between fake and genuine attacks. The KDD Cup99 corpus simulation results showed the model to have substantial performance and respectable levels of precision.

In the HIDE model [14], NNs are used in addition to pre-processed statistical values to classify the movements. In this research, various classification techniques are compared and examined. These approaches consist of BP, RBF, Fuzzy-ARTMAP, and PBH. The PBH and BP models demonstrated noticeably higher levels of detection accuracy when compared to the outcomes produced by the overwhelming majority of the other models. [15] Traffic flow classification is carried out using a method based on detecting anomalies.

A 96% RBF neural network-based classifier detection rate was determined empirically using the UCLA dataset. Fuzzy clustering methods, artificial neural networks, and evolutionary algorithms have all been used to study the difficulty of detecting Distributed Denial of Service assaults.

The study mentioned above [16] created an original model incorporating multilayer perceptions, back propagation, and SVMs. This model was put forth as a solution to the problem and includes sub-models like packet gathering and data pre-processing. A 90.78 percent success percentage in experiments was noted. An overview of both a conventional DoS assault and a distributed DoS attack that could be carried out in the cloud is provided by the authors of the article [17]. Any attack has the potential to disrupt things significantly.

They also suggest defence mechanisms, safety measures, and technical safeguards. Additionally, they discussed a wide range of difficulties and problems associated with preventing DoS attacks in a cloud setting. This highlights the gravity of the issue raised by DDoS attacks on cloud computing, the range of solutions that can be used to address it, and the potential for additional study into mitigating and preventing these attacks.

A formal paper [18] establishes the notion of security known as "non-malleability" under selective opening attacks (NM-SO security). They also examine the connections between the different NMSO security concepts and those between NMSO security and conventional SOA security, as well as non-malleability and conventional security concepts.

Refer to reference [19] to learn more about the spread denial of service attack carried out using a botnet against the application layer of the web server. There are descriptions of events from different parts of the world and financial losses from well-known businesses and official websites; this indicates that extra caution should be used and that more investigation should be done to evaluate the situation fully. [20] suggested an efficient and secure algorithm for solving large systems of linear equations, which is essential and time-consuming across many engineering fields.

The proposed approach can manage non-singular dense matrices, making it sufficiently general to be applied to even the most complex adversary models (a.k.a. fully malicious model). The algorithm used here is more effective and accurate, and the used approach is the current industry gold standard in terms of checking capabilities.

This model type usually ignores the heterogeneity of the features trained into the database. Additionally, malicious traffic flows hint at the consistency of the results obtained for the variables under consideration. The findings indicate drawbacks to expanding the training pool and lowering the approaches' accuracy.

2.2 Analysis of Traffic Flow

An anomaly-based methodology for analyzing traffic flows is proposed in the article [21]. The article's authors predicted [22] that the model may identify DDoS attacks by excluding the offending nodes if traffic flows were clustered according to their IP source addresses. They believed that this was the situation. The study's strength rests in its ability to gather numerous unique sets of attributes despite the high false-positive rate that was seen.

When compared to other recent works on the same topic, this made the study stick out as being very effective. The study in [23] focused heavily on multi-layered artificial neural networks (ANNs) to identify and thwart DDoS attacks. As an additional

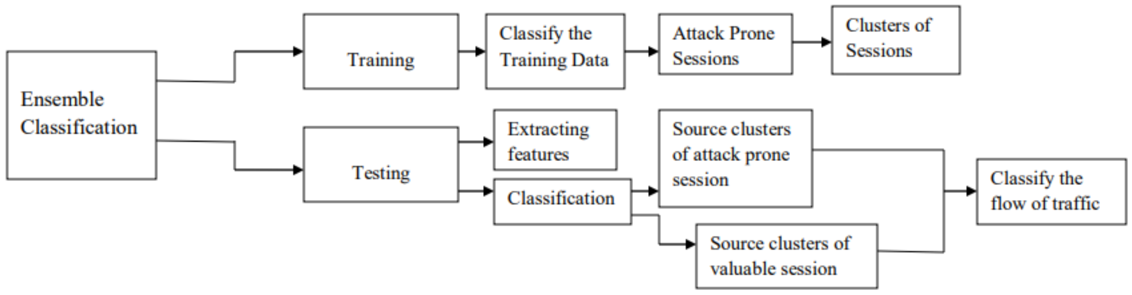

Ensemble Classification
Training
Classify the Training Data
Attack Prone Sessions
Clusters of Sessions
Testing
Extracting features
Classification
Source clusters of attack prone session
Source clusters of valuable session
Classify the flow of traffic

Fig. 1. Operational flow of the proposed model3.1 KS-Test

bonus, traffic flows have been correctly classified as legitimate or malicious using machine learning in combination with NB's classification algorithm [24].

According to a recent study, many academics and researchers praise the value of ensemble learning methods for traffic flow analysis. In the works discussed in [25–27], multi-labeled network flows that reflect various types of real-world traffic are categorized. These compositions rank among the most significant of all time.

Furthermore, the models incorporate a predictor at the kernel level. The simulations showed that the Kernel-based approach [28] offered the highest accuracy when tested on the NSL-KDD corpus. [29] incorporates an adaptive neuro-fuzzy inference method to differentiate between normal and vulnerable traffic flows.

The technique was used to train various models, each with its own data collection. The findings from each classifier were combined in the final phase after each had undergone its own set of tests. The model depended on several static corpora, and the evaluation results carried out using dynamic, practical test corpora were unacceptable, so it has been determined that the experimental study's findings are meaningless. As was already mentioned, all of the versions share this flaw.

The traffic flow characteristics on the network are not accurately reflected by the features used to train classifiers. The bio-inspired flood IDS was trained using the new method, which included adding a new attribute set to enhance this method. The Cuckoo Search algorithm was the foundation for its creation due to its binary classification capabilities. In the empirical analysis of the method, a precision rate of 95+% and a FAR of 4.5+% were discovered.

However, only a corpus of 7100 sessions, all of which consisted of identical transactions, were used in the simulation study. A review of recent studies shows that the success of machine learning techniques depends on selecting the appropriate features and applying the appropriate values during the learning phase. Additionally, the number of datasets increases with the availability of more information.

There are also high FAR rates in traffic analysis methods. This article attempts to categorize particular training corpora into categories based on the diversity of the distribution of their contents.

All classification models taken into account as anensemble component are additionally trained using information from the relevant cluster sets. The suggested solution stands out compared to other contemporary models due to its distinctive application of the metrics and standard learning methodology used in contemporary attribute selection.

As the previous paragraphs show, modern models heavily depend on static databases to determine detection rates. However, this approach uses dynamic databases that combine actual and simulated network attacks. The final step is to select one of these dynamic data repositories. Based on the survey, there are several issues with existing methods in attaining high attack prediction and accuracy.

Hence, this study proposes the ensemble classification model for DDoS attack detection in traffic flow streams.

ISSN 2007-9737

3 Defence of DDoS Attack in Traffic Flow Streams

3.1 KS-Test

In supervised learning, each input is linked to an output label, allowing the system to learn from labelled data. Ensemble classifiers enhance the model's overall accuracy and resilience by combining the predictions of several basic classifiers.

Combining several classifiers, an ensemble classifier aims to minimize the error (misclassification rate) caused by an inadequate classifier. The fundamental premise is to get the predictions of many classifiers using the initial data and then merge their predictions to form a robust classifier.

Figure 1 shows the operational flow of the proposed model. Devices or sensors connected to the network gather data on traffic flow. These data sets often include information regarding packet the aders, including source and destination IP addresses, port numbers, protocol type, and timestamps.

Several relevant features are collected from the data on the flow of traffic. These features may include but are not limited to packet rates, packet sizes, inter-arrival periods, protocol distribution, and other features.

The labels applied to the data indicate whether each network flow indicates normal activity or a DDoS attack. This tagging may be implemented manually or automatically by making use of attack patterns that are already known. An ensemble classifier, which is made up of numerous ensemble classifiers, is trained with the help of the labelled data.

Each ensemble classifier can make use of a different algorithm or feature ensemble. The ensemble classifier uses methods such as majority voting and weighted averaging to aggregate the predictions made by its basis classifiers.

To assess the effectiveness of the trained ensemble classifier in identifying distributed denial of service attacks, it is tested on new traffic flow data that it has not seen before. Attributes like accuracy, precision, recall, F1-score, and ROC curves are used to assess the ensemble classifier's effectiveness. Because of this, its ability to differentiate between regular traffic and distributed denial of service assaults can be evaluated more accurately. The study [30] and potential future research may use single or multiple bio-inspired algorithms to find anomaly-based diverging flood detection.

The strategy outlined in this chapter also has the potential to be improved further to recognize different types of attacks that might be met in networks, like flash crowds. The KS-Test is used to assess the mathematical similarity of two vector distributions. The formulation of this assessment is as follows: Incorporate the sum of $v_1$and v_2's vector values $\|v_1\|$, $\|v_2\|$ into your calculations. A sequence of given vectors can be used with the following equation 1 to calculate the cumulative ratio of its entries:

$$\begin{cases} cr = 0 \\ \bigvee_{j=1}^{|v_i|} \{e_j \exists e_j \in v_i\} \, begin \\ cr = \dfrac{e_j}{\|v_i\|} + cr \\ CR_{v_i} \leftarrow cr \\ end. \end{cases} \tag{1}$$

The procedure is performed many times overall because each supplied vector vi has its cumulative ratio elements computed several times. Calculates the ratio of the element ej to the weighted average of the goal vector v_i.

The sequence of an element that occurs in the supplied vector v_i as element e_j is used to determine the iteration process for each vector. The cumulative ratio cr is then applied to the sum of these values.

The cumulative ratios of every element in the input vector are then added to create a collection of numbers known as CRvi. The $CR_{V_1}\, CR_{V_2}$ sets are tested and saved to identify cumulative ratios, and the ratios of the components present in the target vectors v_1, v_2 are analyzed.

The KS-test then reveals the exact distance between the two vectors' cumulative ratios, indicated by a similar number in the two sets CR_{V_1}, CR_{V_2}, and preserved in the set $diff(CR_{V_1} \leftrightarrow CR_{V_2})$. This is demonstrated in Equation 2:

Table 1. How the algorithm's code is employed

Code	Representation
$SL=\{s_1,s_2,\ldots s_{\|SL\|}\}$	Based on the session starting time, the list is sorted in order.
$TSL \leftarrow SL$	Replicate the sorted sessions list.
$b\ (s_i)$	the session $s_i$start time
$e\ (s_i)$	the session $s_i$ending time
scl_j	the forming of the jth cluster, with j = 1 to start

$$diff(CR_{V_1} \leftrightarrow CR_{V_2}) \leftarrow abs(cr_i - cr_j). \quad (2)$$

The maximum number of the d-stat, or distance measuring statistic $diff(CR_{V_1} \leftrightarrow CR_{V_2})$, is also displayed. The combined values of v_1 and v_2 for the target vectors-critic at the probability degree VIZ threshold are then displayed using the KStable; depending on the TPR distribution, common values for this threshold are 0.01, 0.05, and 0.1. The distributions of the vectors v_1, v_2 are either similar or different from one another, respectively, when the d-stat is greater than the d- critic.

3.2 Clustering the Sessions

The target traffic flow sessions are organized coherently in this article. We will also examine how each possibly vulnerable and desirable subset of sessions is handled separately. We organized the sessions from the earliest to the latest start time based on label traffic patterns and the total number of sessions.

The process entails sublists of sessions being grouped per a predetermined scheme, carried out repeatedly until the complete session list is in chronological order.

The session that has been enrolled in the cluster in ascending order of the listed session is used as the cluster centroid scli after the cluster formation process has been repeated several times for each cluster.

Sessions are sent to the cluster sclito sort when their start times coincide with the centroid's end time. Select the scli cluster centroid with the most sessions with a " Concluded " time stamp within the target scli cluster.

When the centres of each scli cluster remain constant throughout time, this verifies the specified order of cluster creation.

If the newly chosen centroid is not part of the same cluster as the one that was previously used, the sessions are rearranged so that their start time is earlier than their end time relative to the centroid of the cluster scli. The method involves iteratively selecting a new centroid and moving sessions to the target cluster until the centroid is secure. The idea behind the sessions clustering algorithm is depicted in Table 1.

3.3 Usage of Characteristics

The distributional similarity of the following inputs indicates how closely the session groups are related. The following distribution similarity data can be used to demonstrate the session cluster consistency.

1 A vector whose length is proportional to the size of each cluster is used to symbolize the initial intervals of a session. All groups keep tabs on the overall session count. How the cluster graphically depicts the session start time vector:

$$\overset{|C_i|-1}{\underset{j=1}{\forall}} \left\{ sbi(C_i) \leftarrow \left(bt(s_{j+1}) - bt(s_j) \right) \right\} \quad (3)$$

Intended total sessions present in cluster C_i, in the sequence given $bt(s_j)$, $bt(s_{j+1})$, take the difference in seconds between session s_j and $s_{j+1}$and append it to vector $sbi\ (C_i)$.

Algorithm 1: Sessions grouping algorithm

```
Loop 1: $while(|TSL|>0)$ Begin // whereas $TSL$ is not vacant
```

```
    $c_j = \{s_1 \exists s_1 \in TSL\}$ // take the orderd epicted the first session in list $TSL$ as centroid of the $j^{th}$ cluster $scl_j$
      $scl_j \leftarrow c_j$
      $nc_j \leftarrow c_j$
      Loop 2: $\underset{i=1}{\overset{|TSL|}{\forall}} \{s_i \exists s_i \in TSL \wedge s_i \neq c_j\}$ begin
          $if(b(s_i) < e(c_j))$ begin
              $scl_j \leftarrow s_i$
              $if(e(s_i) > e(c_j))$ begin
                  $nc_j \leftarrow s_i$
              End
          End
      End
      $if(c_j \neq nc_j)$ begin
          $c_j = nc_j$ // the cluster $scl_j$ new centroid
          $scl_j = null$ // the cluster becomes nil
          Go to loop 2
      End
      $else\,if(c_j \equiv nc_j)$ begin
          $TSL = \{TSL \setminus scl_j\}$ // the cluster $scl_j$ entries are detached from the sorted list $TSL$
          $j = j + 1$
      End
End
```

2 Each cluster's completion time is represented by a vector $sbi\,(C_i)$ of size $|C_i| - 1$, and this vector represents the session's Completion Intervals. The number of sessions in each cluster is represented by the symbol $|C_i|$. In this example, the session intervals for cluster C_i are represented by the vector $sbi\,(C_i)$, as shown below:

$$\underset{j=1}{\overset{|C_i|-1}{\forall}} \left\{sci(C_i) \leftarrow \left(abs\left(et(s_{j+1}) - et(s_j)\right)\right)\right\} \tag{4}$$

The whole number of sessions$et(s_j)$, $et(s_{j+1})$ in cluster C$_i$ is represented by the vector sc_i, and the exact variance between the session s_j and s_{j+1}completion intervals is transferred to $sci(C_i)$.

3 Begin $pbi(C_i)$ is a vector with the dimensions $|P(C_i)| - 1$ that represents the timestamps of Page Access for a given collection of pages */P(C$_i$)/*. Even if multiple sessions are active, the total number of pages is arranged from earliest to latest according to the time of first access. $P(C_i)$represents the sum of all pages in cluster $C_i$and is written as a mathematical expression. The shape of the vector $pbi(C_i)$ is depicted below:

$$\underset{j=1}{\overset{|P(C_i)|-1}{\forall}} \left\{pbi(C_i) \leftarrow \left(bt(p_{j+1}) - bt(p_j)\right)\right\} \tag{5}$$

By using the cluster *C$_i$* and its total number of pages $P(C_i)$, one may correctly reorder the pages p_j, p_{j+1} start timings $bt(s_j)$, $bt(s_{j+1})$on the vector $pbi(C_i)$.

4 For each source cluster C_i , the session bandwidth utilization is represented by the vector $bwc(C_i)$, which is of size $|C_i|$. In this case, the "source cluster C_i " displays a total of $C_i$sessions; the notation "C_i " denotes this total. You can tell how much bandwidth a cluster uses by looking at the total bandwidth consumption for all requests in that session. Each session's bandwidth consumption for the specified source cluster C_i is represented in the following visualization by the vector $wc(C_i)$.

Algorithm 2: Bandwidth consumption for Session-level

```
$\underset{j=1}{\overset{|C_i|}{\forall}} \{s_j \exists s_j \in C_i\}$
Begin // entire quantity originating from this cluster
    $bwc(C_i) \leftarrow \sum_{k=1}^{|s_j|} \{bw(p_k) \exists p_k \in s_j\}$
    // total use of the given bandwidth $bwc(p_k)$
For the resulting session S$_j$, the vector $bwc(C_i)$ is updated with links to all relevant pages P$_k$.
End
```

3.4 Similarity of Cluster Analysis

The first step in bringing this model into action is to sort the resulting clusters so that each cluster's total number of sessions is displayed in descending order. In following this technique, you are at the beginning. The steps that are taken to ensure that clusters are indeed comparable after they have been grouped are as follows:

The steps in determining the degree to which each cluster *ts*ᵢ is like cluster r_s are depicted below.

Remember the frequency of each trait appearing in the ts_i, r_s clusters.

i. The KS-Test is used to determine the distribution type between the two vectors. The attribute similarity score between the vector and its corresponding clusters ts_i, r_s is 1 if and only if the vector and 0 is clusters share a similar distribution.
ii. Based on the provided average similarity measure of the ts_i, r_s record sets, we may calculate the distribution similarity ratio of the r_s cluster that is connected with ts_i.
iii. The cluster group with a similarity ratio of one is found once the similarity score is obtained.
iv. Table 5.2 shows the formula for calculating the distributional similarity of one cluster to another.
v. The vector value presented for relevant features is then taken for each cluster session, and this set $A = \{a_1, a_2, \ldots . a|A|\}$ comprises all such features.

Algorithm 3: Assessment of Cluster similarity

```
$\forall_{i=1}^{|T|}\{ts_i \exists ts_i \in T\}$ Begin
// each respective group
    $\forall_{j=i+1}^{|T|}\{ts_j \exists ts_j \in T\}$ Begin
    // for every single one of the other clusters tsᵢ besides the cluster
        $\forall_{k=1}^{|A|}\{a_k \exists a_k \in A\}$ Begin
        // corresponding to everyone of the features
            $v_1 \leftarrow ts_i(a_k)$
            $v_2 \leftarrow ts_j(a_k)$
            $sim_{ts_i \leftrightarrow ts_j} += ks\_test($
            // applying KS-test on vectors $t_j, r_j$ that yields 1 if alike, else yields 0
        End
        $dsr_{ts_i \leftrightarrow ts_j} = \frac{sim_{ts_i \leftrightarrow ts_j}}{|A|}$
        // evaluating "the distribution similarity ratio" $dsr_{ts_i \leftrightarrow ts_j}$ amid record sets $ts_i, ts_j$
    End
End
```

3.5 Drift Detection Source Cluster Selection Process

Let's assume that the cluster groups set *CG*= $\{cg_1, cg_2, \ldots cg|CG|\}$ represents the expression described in subsection 3.4. Each cluster group, denoted by the notation $\{cg_i \exists cg_i \in CG \wedge 1 \leq i \leq |CG|\}$, represents the cluster set with the transitive property of equality on distribution similarity.

An illustration of each cluster group considered as a source cluster is shown in Table 3. The provided training corpus is used for parallel training, with the instruction being given to both the complete session partitions and the tagged attack vulnerable partitions.

These are the preparation phases: source cluster detection (i.e., choosing the cluster with the highest session count among clusters group with the distribution similarity based on the equal transitive property); lifespan-based session clustering (i.e., grouping sessions that overlap during the active period); similarity detection using the Kolmogorov-Smirnov test; the classifier is trained using the chosen source groups, and the drift is detected; and The AdaBoost algorithm,

Table 2. The usage of codes in algorithm 3

Code	Representation
Ks_{test}	Kolmogorov Smirnov test
$ts_i ts_j$	These are grouped items.
$Dsr_{ts_i \leftrightarrow ts_j}$	The proportion of comparable distributions
V_1, V_2	These are examples of vectors.

Table 3. The codes used in algorithm 4

Code	Representation
cg_i	*Group cluster*
sc	*Source cluster*
csm	*count session maximum*

Table 4. The traffic flow statistics were generated

	Flushing out specifics on the escalating load	Normal workload
Session Count	47263	43484
Consumption of Bandwidth	65230 MB	42108 MB
Requests count	725695	726198
Sources count	732	518

which depends on the throughput of ensemble classification, is one of the paper's innovations.

As stated in subsection 3.3, this research aims to train each classifier using an ensemble of classifiers with optimal properties related to traffic flow. As a result, the suggested method can successfully train any appropriate classifier for use with various classifiers.

4 Experimental Study

This section reviews the procedures needed to establish a dataset, develop a strategy, set up an experiment, and compare and contrast the outcomes to determine the strategy's efficacy.

4.1 Dataset on DDoS Attack

When assessing attack defence models against DDoS attack record datasets, such as NSL-KDD [29], it is crucial to keep in mind that these datasets contain requests that have been repeatedly iterated, increasing the risk of false positives. This is because these datasets include requests that were created automatically. [24]

This result is necessary, given the evaluation's nature. The three openly accessible databases from DARPA/Lincoln Labs are KDD [31], CAIDA [32], and [33]. The DAPRA datasets span several weeks of network activity and contain data from a simulated Air Force network.

These files contain no sophisticated attacks and have not been altered in any way. According to the study [34], DARPA has distorted the data. Simply sending a user request will grant the user access to the CAIDA dataset. Information about DDoS that happened in 2007 can be found in this CAIDA.

The DDoS attack dataset that CAIDA has only lasted for an hour and includes traffic records from hypothetical situations. Public datasets are unavailable during DDoS attacks because they expose private network information and foreseeable user behaviour. Consequently, we.
cannot make these datasets available to the general public. A data leak could result in legal issues and risks to the general public, the organization, and the service suppliers. When applied to simulated data, the outcomes of a detection technique are not accurate. Consequently, it is impossible to compare these results to those of the real-time dataset. The transmission of laboratory data deviates from the anticipated amount of traffic on the upstream routers. According to the study's findings [35], it is impossible to generalize from tiny data samples to more extensive databases.

Table 5. Results of the BIFAD experiments

Absolute-time intervals Count	2046(1256:RTF, 790:RTN)
measuring intervals in absolute time	674 (354:RTF, 320:RTN)
Intervals of Absolute Time for Physical Training	1374(749:RTF, 625:RTN)
Representations of records that are wholesome	279
Representations of a record's vulnerability	326
Salubrious records that have been correctly noted (TN)	273
Records from Attack Prone have been referred to as salubrious (FN)	34
Highly vulnerable records have been identified (TP)	296
Salubrious records have been found in attack-prone databases (FP)	42
True positive rate(TPR)	0.923755389
True negative rate(TNR)	0.904636644
Positive predict value	0.892151616
Accuracy	0.898386401
FAR	0.185749276

Table 6. The statistics of input given and outcomes achieved from NFBoost

Overall Salubrious	725183
Overall Attack-Prone	734685
Detection of Positive Results	264377
Detection of Negative Results	253528
Training for Positive Results	520562
Training for Negative Results	520762
TP	250968
FP	31303
TN	232114
FN	32518
TPR	0.892
TNR	0.8946
predict positive value	0.8979
accuracy	0.8997
FAR	0.1648

We evaluated the outcomes of requests produced by DDoSIM, Tor's Hammer, and JMeter. A traffic-generating tool called Tor's Hammer generates lots of traffic while allowing users to publish at their own pace. Consequently, many weak servers are brought down even if they only

Table 7. Data on the construction and selection of ECDD source clusters

	Count of training sessions	**Group of Clusters**	**a single classifier for each cluster of sessions**	**The number of session clusters**	**clusters of sources**
from transaction logs of DDoS attacks	34507	10	11	46	In a range of 721 to 2036
From Normal Transaction	34674	7	8	62	In a range of 562 to 749

have a few users. Based on empirical data, this program demonstrates that standard Apache/IIS servers up to version 6 can abruptly terminate up to 256 threads, whereas versions after this can be compacted by up to 128 threads.

Using the Apache JMeter load-generating tool, which can produce high and varied loads, the performance of the target server or any other server can be assessed in a static or dynamic situation [36].

JMeter, Tor's Hammer, and DDoSIM collected data on regular and attack-prone traffic flows for an hour. To that end, the system was configured to make requests to various sources at regular intervals of 17.6 ms. For instance,Table 4 shows that the attack-prone and typical requests generated 732 malicious and 518 legitimate requests, respectively.

Testing the suggested Ensemble Classifiers with Drift Detection (ECDD)can demonstrate the significance of the claim made in this study and the range of the proposed technique adds to Bio-Inspired Flood Attack Detection (BIFAD)and describes the ensemble of classifiers NFBoost, both of which offer helpful information.

After evaluating each model's performance, the author chooses ECDD and BIFAD; the BIFAD model was chosen due to the effectiveness of its ensemble classifier method. According to research, the BIFAD, because of its distinctive traffic flow characteristics, outperforms the ECDD model in identifying the wide range of DDoS attacks [37]. The results are also contrasted with those from experiments done on the NFBoost model. NFBoostis the most efficient ensemble classifier using the ECDD classifier training method and features of traffic flow traits.

4.2 Result Analysis

The experiment is improved in that it validates the ECDD model and evaluates the significance of the outcomes from the BIFAD model and the NFBoost model, respectively, using cross-justification metrics like the true negative rate (TNR), the true positive rate (TPR), and the positive predict value.

Results from high-traffic periods following the implementation of BIFAD are shown in Table 5, where they demonstrate an increase in attack detection precision to 0.89 and a decrease in FAR to 0.12.reports that a small-scale BIFAD model experiment resulted in an attack detection accuracy of 0.96 and a false alarming ratio of 0.045.

As a result, keeping detection accuracy with high traffic volumes and a variety of vehicle types is not particularly dependent on the level of detection accuracy of the BIFAD model. Therefore, the detection accuracy level of the BIFAD model is not especially crucial for preserving detection accuracy.

Additionally, Table 6 displays the outcomes of tests conducted using the NFBoost model. This NFBoost model is trained and assessed using requests rather than sessions in a pre-defined traffic pattern. These figures show that the classifier was trained and evaluated on the real traffic flow at a 70:30 split during the experiment.

According to the findings, the false alarm ratio is 0.12, equal to 12% accuracy, and the detection ratio for the transaction state is 0.89, comparable to 89% accuracy.

The report shows the results of an empirical study on a total of 5150 attack-resistant occurrences and 12950 attackable occurrences.

Table 8. The ECDD testing process is provided as inputs

Number of times spent training vulnerable positions to attack	32408
The number of Salubrious training sessions	31653
Participants in healthy training sessions form groups	63
groups of possible victims of abuse receiving training	45
Testing for Negatives	10613
Testing for Positives	11548
clusters formed by treatment activities with the objective of evaluation (total negatives)	31
Attack-prone training sessions are used to form groups (total positives)	32
Attack-prone groups were incorrectly depicted (False Positives)	4
Groups with a high potential for attack (True Positives)	32
Groups that have been misrepresented as negative (False Negatives)	4
Correctly salubrious groups are represented (True Negatives)	30
Accuracy	0.964673
TPR	0.96
TNR	0.959342
FAR	0.09
predict positive value	0.96

According to the research, the NFBoost model had a detection accuracy of 96% and a false alarm rate of 2.8%.

Results from the previous work on the NFBoost (an ensemble classifier model imported from the source) show an unusually high degree of variance compared to the current experimental study.

The practical analysis of NFBoost yields dismal results. Regarding the flow of created traffic, the training method for NFBoost had many examples and a high degree of variation among those examples, both of which added to the algorithm's subpar performance. The ECDD model is also discussed here, and the results obtained are shown in Tables 7-8 and Figure 2.

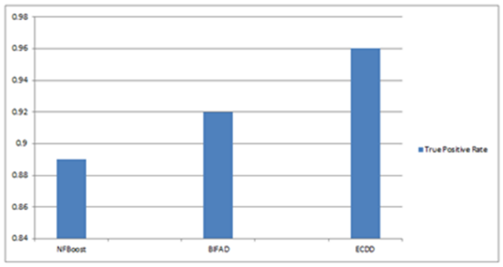
The results of experimental studies used to assess the effectiveness of the ECDD paradigm are shown in Table 8. Compared to the experimental research results, these findings showed that the ECDD model performed better than the BIFAD and NFBoost models in identifying attacks. We discovered that the success rate was 96%, and the erroneous alarm rate was 8%.

This showed the effectiveness of the ECDD model in terms of performance in detecting attacks. The animals have spoken, and BIFAD and NFBoost have an attack detection rate of over 96%.

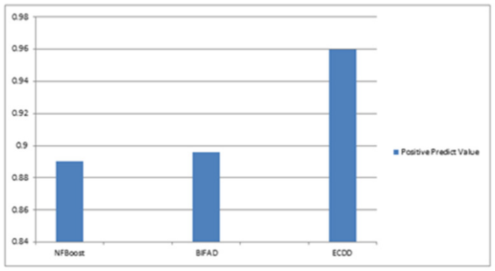
The performance, however, progressively deteriorates when the classifiers are trained on a large volume of traffic and a variety of request types. Using cutting-edge traffic flow features, BIFAD equals the NFBoost ensemble classifier's 96% identification success.

This is made feasible by the distinctive features of the traffic flow at BIFAD. This implies that the two most current models cannot handle large amounts of traffic with various request types.

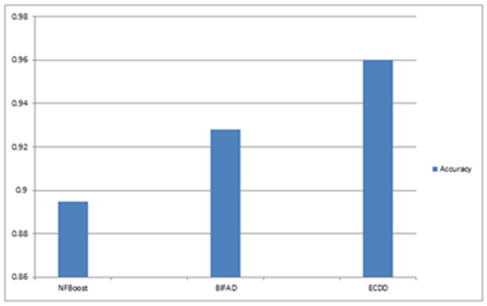
This results in a linear and reasonably quick ECDD processing time for the suggested model.



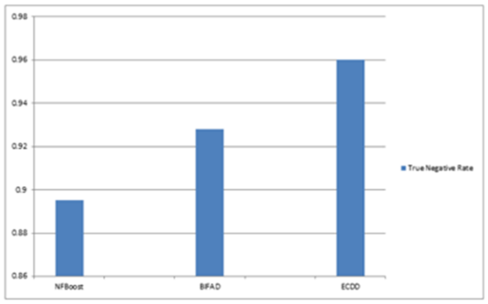
a) True predict value



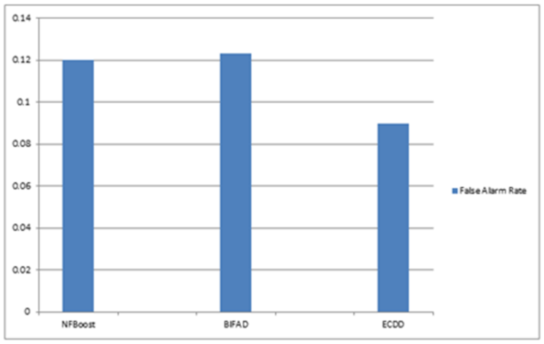
b) Positive predict value



(c) Accuracy



(d) Truenegative rate



e) False alarm rate

Fig. 2. The experimental study's presentation statistics are shown in the graphs.

5 Conclusion

The computer networks sector has managed the anticipated influx of user requests flooding the internet service. It is now crucial to continuously work on enhancing server performance. Requests that a trustworthy user verifies and those marked by an attacker are the same to the system.

The security of contemporary internet-based services that house servers for various domain platforms, including those used for entertainment and commerce, is seriously threatened by distributed denial-of-service, or DDoS, attacks.

These services offer clients a selection of places ideal for personal and professional activities. It's crucial to practise the defence mechanism often with heavy traffic and various distribution patterns. The benchmark models used today have so far produced ambiguous findings.

The difference in traffic trends at each node in the distribution network led to inaccurate attack detection. The performance of the current models is compared to a set of benchmark models, which notably feature the BIFAD and NFBoost, to ensure they can handle the massive traffic flow through significant distribution variety qualities. This is done to make sure the current system is current. As a result, experimental research comparisons with other models revealed the scalability and durability of the ECDD model.

Experiments with both cutting-edge and novel models demonstrated that hybrid versions, like BIFAD and ECDD, predicted noticeably better performance to improve the precision of drift and attack detection while reducing the frequency of false alarms.

References

1. **Patil, N. V., Krishna, C. R., Kumar, K. S. (2022).** SSK-DDoS: distributed stream processing framework based classification system for DDoS attacks. Cluster Computing, Vol. 25, No. 5, pp. 1355–1372. DOI: 10.1007/s10586-022-03538-x.
2. **Jain, M., Kaur, G. (2021).** Distributed anomaly detection using concept drift detection based hybrid ensemble techniques in streamed network data. Cluster Computing, Vol. 24, No. 5, pp. 2099–2114. DOI: 10.1007/s10586-021-03249-9.
3. **Granato, G., Martino, A., Baldini, L., Rizzi, A. (2022).** Intrusion detection in wi-fi networks by modular and optimized ensemble of classifiers: an extended analysis. SN Computer Science, Vol. 3, No. 4, p. 310. DOI: 10.1007/s42979-022-01191-0.
4. **Munivara-Prasad, K., Rama-Mohan-Reddy, A., Venugopal-Rao, K. (2017).** BIFAD: Bio-inspired anomaly based HTTP-flood attack detection. Wireless Personal Communications, Vol. 97, pp. 281–308. DOI: 10.1007/s11277-017-4505-8.
5. **VivinSandar, S., Shenai, S. (2012).** Economic denial of sustainability (EDoS) in cloud services using HTTP and XML based DDoS attacks. International Journal of Computer Applications, Vol. 41, No. 20, pp. 11–16. DOI: 10.5120/5807-8063.
6. **Loukas, G., Öke, G. (2010).** Protection against denial of service attacks: A survey. The Computer Journal, Vol. 53, No. 7, pp. 1020–1037. DOI: 10.1093/comjnl/bxp078.
7. **Linda, O., Vollmer, T., Manic, M. (2009).** Neural network based intrusion detection system for critical infrastructures. 2009 international joint conference on neural networks, IEEE.pp. 1827–1834. DOI: 10.1109/ IJCNN.2009.5178592.
8. **Apale, S., Kamble, R., Ghodekar, M., Nemade, H., Waghmode, R. (2014).** Defense mechanism for DDoS attack through machine learning. International Journal of Research in Engineering and Technology, Vol. 3, No. 10, pp. 291–294. DOI: 10.15623/ijret.2014. 0310045.
9. **Vijayasarathy, R., Raghavan, S. V., Ravindran, B. (2011).** A system approach to network modeling for DDoS detection using a Naive Bayesian classifier. Third International Conference on Communication Systems and Networks, IEEE, pp. 1–10. DOI: 10.1109/ COMSNETS.2011.5716474.
10. **Lu, K., Wu, D., Fan, J., Todorovic, S., Nucci, A. (2007).** Robust and efficient detection of DDoS attacks for large-scale internet. Computer Networks, Vol. 51, No. 18, pp. 5036–5056. DOI: 10.1016/j.comnet.2007. 08.008.
11. **Pan, W., Li, W. (2005).** A hybrid neural network approach to the classification of novel attacks for intrusion detection. International Symposium on Parallel and Distributed Processing and Applications, pp. 564–575. DOI: 10.1007/11576235_58.
12. **Pacheco, J., Benitez, V. H., Felix-Herran, L. C., Satam, P. (2020).** Artificial neural networks-based intrusion detection system for internet of things fog nodes. IEEE Access,Vol. 8, pp. 73907–73918. DOI: 10.1109/ACCESS. 2020.2988055.

13. **Haddadi, F., Khanchi, S., Shetabi, M., Derhami, V. (2010).** Intrusion detection and attack classification using feed-forward neural network. Proceedings of the 2010 Second International Conference on Computer and Network Technology, IEEE Computer Society, Washington DC, pp. 262–266. DOI: 10.1109/ICCNT.2010.28.

14. **Mendonça, R. V., Teodoro, A. A., Rosa, R. L., Saadi, M., Melgarejo, D. C., Nardelli, P. H., Rodríguez, D. Z. (2021).** Intrusion detection system based on fast hierarchical deep convolutional neural network. IEEE Access, Vol. 9, pp. 61024–61034. DOI: 10.11 09/ACCESS.2021.3074664.

15. **Lopez-Martin, M., Sanchez-Esguevillas, A., Arribas, J. I., Carro, B. (2021).** Network intrusion detection based on extended RBF neural network with offline reinforcement learning. IEEE Access, Vol. 9, pp.153153–153170. DOI: 10.1109/ACCESS.2021. 3127689.

16. **Albahar, M. A., Binsawad, M., Almalki, J., El-etriby, S., Karali, S. (2020).** Improving intrusion detection system using artificial neural network. International Journal of Advanced Computer Science and Applications, Vol. 11, No. 6. DOI: 10.14569/ijacsa.2020.0110670.

17. **Gupta, B. B., Badve, O. P. (2017).** Taxonomy of DoS and DDoS attacks and desirable defense mechanism in a cloud computing environment. Neural Computing and Applications, Vol. 28, No. 12, pp. 3655–3682. DOI: 10.1007/s00521-016-2317-5.

18. **Huang, Z., Liu, S., Mao, X., Chen, K., Li, J. (2017).** Insight of the protection for data security under selective opening attacks. Information Sciences, Vol. 412, pp. 223–241. DOI: 10.1016/j.ins.2017.05.031.

19. **Gangadharan, K., Kumari, G. R. N., Dhanasekaran, D., Malathi, K. (2020).** Detection and classification of various pest attacks and infection on plants using RBPN with GA based PSO algorithm. Indonesian Journal of Electrical Engineering and Computer Science (IJEECS), Vol. 20, No. 3, pp. 1278–1288. DOI: 10.11591/ijeecs.v20.i3.

20. **Chen, X., Huang, X., Li, J., Ma, J., Lou, W., Wong, D. S. (2015).** New algorithms for secure outsourcing of large-scale systems of linear equations. IEEE Transactions on Information Forensics and Security, Vol. 10, No. 1, p. 38. DOI: 10.1109/TIFS.2014.2363765.

21. **Barford, P., Plonka, D. (2001).** Characteristics of network traffic flow anomalies. Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement, pp. 69–73. DOI: 10.1145/505 202.505211.

22. **Kalliola, A., Lee, K., Lee, H., Aura, T. (2015).** Flooding DDoS mitigation and traffic management with software defined networking. Proceedings of 2015 IEEE 4th International Conference on Cloud Networking, IEEE, pp. 248–254. DOI: 10.1109/CloudNet.2015.7335317.

23. **Seufert, S., O'Brien, D. (2007).** Machine learning for automatic defence against distributed denial of service attacks. IEEE International Conference on Communications, pp. 1217–1222. DOI: 10.1109/ICC.2007.206.

24. **Berral, J. L., Poggi, N., Alonso, J., Gavalda, R., Torres, J., Parashar, M. (2008).** Adaptive distributed mechanism against flooding network attacks based on machine learning. Proceedings of the 1st ACM workshop on Workshop on AISec, pp. 43–50. DOI: 10.1145/1456377.1456389.

25. **Huang, G. B., Zhou, H., Ding, X., Zhang, R. (2012).** Extreme learning machine for regression and multiclass classification. IEEE Transactions on Systems, Man, and Cybernetics, Part B, Vol. 42, No. 2, pp. 513–529. DOI: 10.1109/TSMCB.2011.2168604.

26. **Srimuang, W., Intarasothonchun, S. (2015).** Classification model of network intrusion using weighted extreme learning machine. 2015 12th International Joint Conference on Computer Science and Software Engineering pp. 190–194. DOI: 10.1109/JCSSE.2015. 7219794.

27. **Fossaceca, J. M., Mazzuchi, T. A., Sarkani, S. (2015).** MARK-ELM: application of a novel multiple Kernel learning framework for improving the robustness of network intrusion detection. Expert Systems with Applications,

Vol. 42, No. 8, pp. 4062–4080. DOI: 10.1016/j.eswa.2014. 12.040.

28. **Raj-Kumar, P. A., Selvakumar, S. (2013).** Detection of distributed denial of service attacks using an ensemble of adaptive and hybrid neurofuzzy systems. Computer Communications, Vol. 36, No. 3, pp. 303–319. DOI: 10.1016/j.comcom.2012.09.010.
29. **Revathi, S., Malathi, A. (2013).** A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection. International Journal of Engineering Research and Technology. DOI: 10.17762/turcomat.v12i11.6333.
30. **Nazario, J. (2008).** DDoS attack evolution. Network Security, Vol. 2008, No. 7, pp. 7–10. DOI: 10.1016/S1353-4858(08)70086-2.
31. **Shin, S., Gu, G., Reddy, N., Lee, C. P. (2011).** A large-scale empirical study of conficker. IEEE Transactions on Information Forensics and Security, Vol. 7, No. 2, pp. 676–690. DOI: 10.1109/TIFS.2011.2173486.
32. **Sommer, R., Paxson, V. (2010).** Outside the closed world: On using machine learning for network intrusion detection. Security and Privacy (SP), 2010 IEEE Symposium on, pp. 305–316. DOI: 10.1109/SP.2010.25.
33. **Behal, S., Kumar, K. (2017).** Characterization and comparison of DDoS attack tools and traffic generators: A review. International Journal of Network Security, Vol. 19, No. 3, pp. 383–393. DOI: 10.6633/IJNS.201703. 19(3).07.
34. **Badve, O. P., Gupta, B. B. (2016).** Taxonomy of recent DDoS attack prevention, detection, and response schemes in cloud environment. Proceedings of the International Conference on Recent Cognizance in Wireless Communication & Image Processing: ICRCWIP-2014, Springer India, pp. 683–693. DOI: 10.1007/978-81-322-2638-3_76.
35. **Saikam, J., Ch, K. (2024).** An ensemble approach-based intrusion detection system utilizing ISHO-HBA and SE-ResNet152. International Journal of Information Security, Vol. 23, No. 2, pp. 1037–1054. DOI: 10.1007/s10207-023-00777-w.
36. **González, F. J., Aguirre-Anaya, E., Salinas-Rosales, M., Miyaji, A. (2023).** Identification of static and dynamic security controls using machine learning. Computación y Sistemas, Vol. 27, No. 2, pp. 581–592. DOI: 10.13053/CyS-27-2-4429.
37. **Chaparro-Amaro, O. R., Martínez-Felipe, M. D. J., Martínez-Castro, J. A. (2023).** Performance of the classification of critical residues at the interface of BMPs complexes pondered with the ground-state energy feature using random forest classifier. Computación y Sistemas, Vol. 27, No. 1, pp. 257–267. DOI: 10.13053/CyS-27-1-4537.

Article received on 28/02/2024; accepted on 15/05/2024.
**Corresponding author is Dutta Sai Eswari.*