A Novel Approach for Detection of DoS / DDoS Attack in Network Environment Using Hellinger Distance Technique

Sudhir Kumar Pandey, Ditipriya Sinha
Department of Computer Science Engineering, National Institute of Technology (NIT), Patna, India
E-mail: sudhirp.phd19.cs@nitp.ac.in, ditipriya.cse@nitp.ac.in

Keywords: DoS/DDoS Attack, Hellinger distance, machine learning, intrusion detection system, NSL-KDD, UNSW-NB15, real time dataset

Received: February 1, 2024

As the popularity of the Internet increases, the number of threats is also growing at a rapid pace. A DoS attack is one such threat. DoS attacks deplete network bandwidth, thereby preventing genuine users from accessing resources. These attacks exhaust the computing resources of their victims without advanced warning. This paper proposes a novel framework that applies the Hellinger distance technique to handle these attacks efficiently. The proposed framework demonstrates promising performance in detecting a diverse range of DoS/DDoS attacks. The approach consists of three phases: data processing, threshold generation, and anomaly detection. We collected traffic data via a created virtual environment and applied feature selection methods such as the Chi-square test, Pearson correlation, and feature importance using a tree classifier during the preprocessing phase. To generate a threshold, we calculated the probability distribution of each profile collected and, after conducting sensitivity analysis, determined the threshold value that provided the best accuracy.We tested the performance of our approach using real network traces. The proposed model achieved an accuracy of 99.41% in detecting DoS attacks. Additionally, this paper analysed and compared our own dataset with the standard NSL-KDD and UNSW_NB15 datasets, and it was observed that our proposed model achieved better accuracy, with 85.29% and 81.42% for the NSL-KDD and UNSW_NB15 datasets, respectively.

Povzetek: Razvit je nov pristop za zaznavanje DoS/DDoS napadov s tehniko Hellingerjeve razdalje. Model vključuje faze obdelave podatkov, generacije praga in zaznavanja anomalij. Testiranje na realnih in standardnih podatkovnih zbirkah (NSL-KDD, UNSW-NB15) je pokazalo izboljšave pri zaznavi napadov.

1 Introduction

Confidentiality, integrity, and availability are the three essential pillars of cybersecurity. DoS attacks aim to drain resources (memory, CPU cycles, and network bandwidth) and render them inaccessible to authorized users, thereby jeopardizing one of the most crucial components of cybersecurity: availability [1]. A DoS attack makes a system or network service unavailable to its intended or legitimate users by flooding servers with enormous traffic. DoS attacks deprive legitimate users of the services or resources they expected from the machine or network servers.While any system or network can be a target of DoS attacks, attackers often focus on high-profile organizations in the IT industry, banking sector, government websites, and similar sectors. Although DoS attacks typically do not result in data leaks or loss of information, they can cost victims significant time and money to recover from the disruption [2].

The cybersecurity techniques of the modern era have evolved to protect internet sites from DoS and DDoS attacks. However, attacks involving multiple systems or botnets remain a significant threat to organizations, governments, websites, and other entities[3]. Companies are continuously working to counter DDoS attacks and reduce their impact.DoS attacks are classified into three categories: application-layer attacks, protocol-based attacks, and volume-based attacks. Protocol-based attacks exploit weaknesses in the network and transport layers of the protocol stack [4]. These attacks deplete resources by targeting servers, firewalls, and load balancers. The two most prominent protocol-based assaults are SYN Flood and Ping of Death. SYN Flood attacks exploit vulnerabilities in the TCP connection sequence. The attacker sends a large number of SYN packets to the target server. When the server responds with a SYN/ACK packet to acknowledge the connection, it does not receive the final ACK signal from the client [5]. As a result, false connections are opened on the server port, waiting for the final handshake from the client. Once all the ports are occupied, the server's functionality is disrupted, preventing any new connections from being established. SYN attacks can be either direct (without spoofing) or involve IP address spoofing. Direct SYN attacks are easier to mitigate. Volume-based attacks are the simplest to generate. These attacks aim to consume all available network bandwidth by flooding the target server with

massive amounts of traffic. Common volume-based attacks include NTP amplification, DNS amplification, UDP flood, and TCP flood attacks [6].

This research utilizes the Hellinger distance technique to analyze denial-of-service (DoS) attacks. DoS attacks that exploit application-level vulnerabilities have recently become an increasing threat to web applications. Unlike network-level DoS attacks, these threats typically consume minimal bandwidth, making them more challenging to detect[7]. For instance, an attack that exploits the fact that an authentication server must perform resource-intensive RSA decryption operations requires only a few megabits per second of bandwidth to take a website offline. Such an attack rate could easily blend in with normal traffic.

Application-layer DoS attacks target web servers and websites by exploiting vulnerabilities through GET/POST floods [8]. The primary goal of these attacks is to crash the web servers, making them a serious threat to web services. Common application-layer attacks include HTTP flood attacks and attacks on DNS servers. One notable attack, Slowloris, enables attackers to take down a server by maintaining multiple simultaneous HTTP connections. Slowloris uses partial requests to keep these connections open for extended periods. As a result, the server's connection pool becomes fully occupied, preventing legitimate clients from establishing new connections. In an HTTP flood DoS attack, the attacker uses seemingly legitimate HTTP GET or POST requests to target a web server or application. Unlike other types of attacks, the attacker does not use erroneous packets, spoofing, or reflection techniques, and the attack requires less bandwidth compared to similar methods.

This attack is most effective when the server or application is forced to allocate the maximum amount of resources in response to each request. To maximize efficiency, attackers often use botnets. An HTTP flood attack sends data slowly but fast enough to prevent the server from timing out and closing the connection. The mechanism of a DoS attack is illustrated in Figure 1.

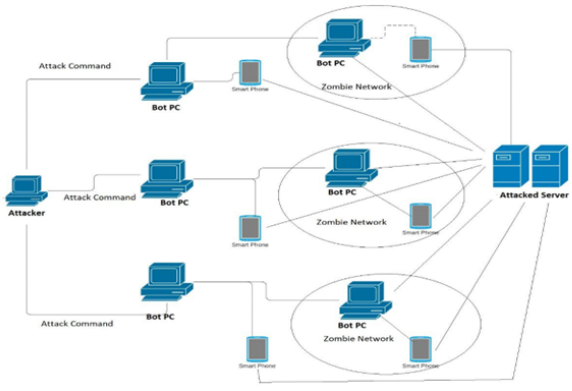


Figure 1: Working mechanism of DoS attack

DoS attack techniques can be categorized into three types, as shown in Figure 2: spoofing, resource exhaustion, and vulnerability exploitation.

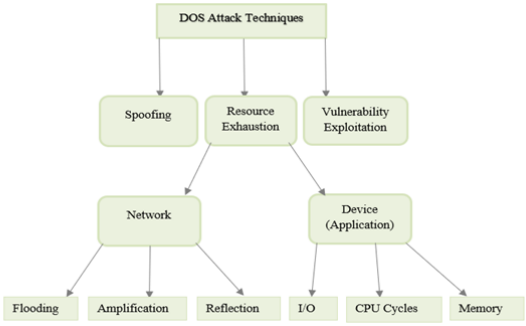


Figure 2: Taxonomy of DoS attack techniques

IP spoofing is one of the most commonly used techniques for launching DoS attacks. In simple terms, IP spoofing occurs when an attacker sends a request to the target with a fraudulent source IP address in the IP header. This is possible because devices can be assigned any IP address. Flooding, amplification, and reflection attacks all use IP spoofing as a fundamental tactic.

In flooding attacks, the attacker exploits the TCP three-way handshake process by sending a large number of SYN packets to the target server with spoofed source IP addresses. The server then attempts to send ACK signals to the nonexistent IP addresses, consuming resources and preventing the server from processing legitimate connection requests.

Amplification and reflection attacks commonly exploit applications that use the UDP protocol. The attacker sends small UDP packets to vulnerable UDP servers with spoofed source IP addresses. These servers then send response messages to the victim's system at the spoofed IP address. This strategy is known as a reflection attack. An amplification attack occurs when the server's response is significantly larger than the initial fraudulent request, causing an overwhelming amount of data to be sent to the target.

While some resource exhaustion attacks involve spoofing, we distinguish between the two strategies because not all resource exhaustion attacks use spoofing, and not all spoofing is employed for resource exhaustion. Resource exhaustion methods rely on any tactic that can saturate available bandwidth, CPU, memory, or system bus throughput on end devices—not just spoofing. These tactics may include overloading standard applications and services on portable devices.

Vulnerability exploitation can be understood from the perspective of protocol design, but in our taxonomy, we focus on flaws or vulnerabilities in application binaries. A DoS attack can originate from a single device or multiple devices using either spoofed or legitimate IP addresses.

Figure 3 illustrates DoS attacks from the perspective of the source. The majority of modern DoS attacks employ a combination of spoofed and legitimate IP addresses. The most common type of attack is the Distributed Denial of

Service (DDoS) attack, which is one of the most dangerous and destructive threats on the Internet. [9], [10].

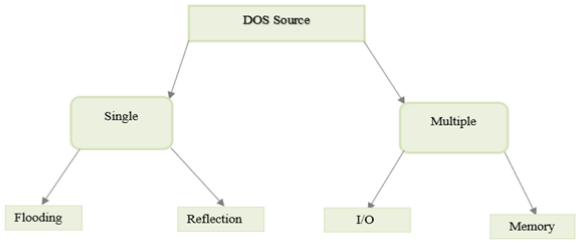


Figure 3: Different DoS sources

1.1 Contribution highlights of the paper

The contribution of my proposed work can be summarized as below:

1. A hellinger distance-based DoS attack detection model is proposed. Proposed model consists of three phases: (a) Data Processing Phase (b) Threshold generation phase (c) Anomaly detection phase.

2. The classifier was developed using a raw byte stream of real-time network traffic that was recorded.

3. The performance of the model is evaluated on the traditional machine learning algorithms such as Logistic Regression, Naïve Bayes, SVM and KNN, using different performance metrics.

4. The Model is evaluated with our generated real time dataset, standard NSL-KDD dataset and UNSW-NB15 Dataset. It is shown that the proposed model gives higher accuracy compared to existing approaches.

5. The proposed model is also analyzed for threshold using sensitivity analysis based on the different threshold value and window size trade off. .

1.2 Outline of the paper

The organization of the paper is as follows: Section 2 reviews recent contributions by various authors in the field of DoS attack detection. Section 3 outlines the initial concepts required for designing the proposed work. Section 4 provides a comprehensive overview of the proposed framework. Section 5 details the dataset collected for identifying DoS attacks. Section 6 explains several performance metrics and presents the results analysis. Finally, Section 7 concludes the paper by discussing the advantages and limitations of the proposed framework.

2 Literature survey

DoS attack detection has recently gained significant attention. The most common types of DoS attacks aim to deplete the target system's network bandwidth, CPU cycles, or memory, making services inaccessible to legitimate users [11]. Various techniques have been proposed, including behavioral and graph-based approaches, which are implemented at the packet and kernel levels. Machine Learning and Deep Learning methods also play a crucial role in preventing DoS and DDoS attacks. This section reviews relevant work that has been done in identifying DoS attacks.

The approach proposed by [1] treats traffic records as images and frames DoS detection as a computer vision problem. They developed a multivariate correlation analysis method to accurately represent network traffic logs and convert them into images. In their proposed DoS attack detection system, these images are used as the observed objects, with detection based on the Earth Mover's Distance (EMD), a widely used dissimilarity metric. EMD performs cross-bin matching and provides a more accurate assessment of distribution dissimilarity compared to other well-known metrics, such as Minkowski distance, Lp, and X^2 statistics. These distinguishing characteristics enable their system to effectively detect attacks.

They conducted ten-fold cross-validation on the EMD-based detection system using the KDD Cup 99 dataset and the ISCX 2012 IDS Evaluation dataset. The results, presented in the system evaluation section, show that their detection system achieves a detection accuracy of 99.95% on the KDD Cup 99 dataset and 90.12% on the ISCX 2012 IDS Evaluation dataset, with a processing capacity of approximately 59,000 traffic records per second.

[12] introduced an intelligent DoS detection system that includes modules for data generation, feature ranking and creation, and training and testing. The proposed framework was evaluated in real-world IoT attack scenarios and demonstrated greater accuracy than existing classification algorithms. [13] presents a machine learning approach for detecting flooding Denial of Service (DoS) attacks in IEEE 802.11 networks. The dataset used in this study was created in a computer lab by the authors, consisting of 40 computers, with seven identified as attackers to launch DoS attacks. Each legitimate node is connected to one of the five available access points (APs). The dataset was divided into two parts: 66% for machine learning training and 34% for testing. Using the WEKA application, the authors applied six different classification ML algorithms in sequence: SVM, Naive Bayes, Naive Bayes Net, Ripple-Down Rule Learner (RIDOR), Alternating Decision Tree, and Adaptive Boosting (AdaBoost). Based on accuracy and recall metrics, the empirical data shows that AdaBoost outperforms the other methods.

In a wireless video sensor network, [14] proposed an optimized deep neural network-based DoS attack detection method. This approach was compared to the RAS-HO, TMS, and SVM-DoS methods. The results of the tests sug-

gest that optimization techniques can enhance the performance of the learning process. Additionally, it was found that feature selection reduces the dataset's dimensionality. Parameters are selected using an adaptive particle swarm optimization technique. The efficiency of the approach will be evaluated based on metrics such as packet transmission ratio, energy consumption, latency, network length, and throughput.

[2] introduced a method for detecting SIP-based Denial of Service attacks using a Dual Cost Formulation of Support Vector Machine (SVM). They created SIP traffic in their lab and empirically evaluated the performance of various classifiers. The SVM performed well compared to other classifiers, but it did produce some false positives and false negatives according to the experimental data.The proposed SVMBoost method was tested on both high-rate and low-rate message flooding datasets. SVMBoost achieved a detection accuracy of 99.9%, with a false positive rate of 0% and a false negative rate of 0.27%. This method outperformed previous algorithms in high-rate flooding detection, showing a maximum improvement of 35.97% a minimum improvement of 0.28%, and an average improvement of 4.45% in low-rate flooding detection.

Based on machine learning approaches, [15] developed a DoS attack detection system on the source side in the cloud. This solution uses statistical information from both the cloud server's hypervisor and the virtual machines to prevent network packets from being pushed out to the external network. Their findings indicate that more than 99.7% of four different types of DoS attacks can be effectively identified. [3]proposed using machine learning (ML) and neural network (NN) algorithms for DoS attack detection. They focused on application layer DoS attacks and preferred these methods for detecting transport and network layer DoS attacks. The study utilized the CICIDS 2017 dataset, the most recent DoS attack dataset. The dataset was divided into multiple splits during the experiments, and the optimal split was determined for each method. The results showed that Random Forest (RF) provided better performance compared to Multi-Layer Perceptron (MLP).

[5] proposed an empirical study on slow HTTP DoS attacks and their detection. They made two key contributions to this work. First, they conducted an empirical investigation on several HTTP servers to assess their vulnerability to slow HTTP DoS attacks. Second, they proposed a technique for detecting these attacks. The suggested anomaly detection method calculates the Hellinger distance between two probability distributions generated during the training and testing stages.

[16]introduced a method for detecting DoS attacks using a multilayer Deep Belief Network (DBN), consisting of multiple Restricted Boltzmann Machines (RBMs). In this approach, RBM training occurs early in the learning process. The features learned by the RBMs are then used as input for training the next layer of RBMs in the DBN stack. The effectiveness of the DBN approach was evaluated using the KDD-CUP 1999 dataset, where it demonstrated superior detection accuracy compared to SVM and ANN approaches.

[7] developed a trust-based DoS attack detection system for secure data transfer in wireless sensor networks. This paper introduces an effective trust-based module designed to detect denial of service attacks, including selective forwarding and flooding attacks. The proposed technique extracts and estimates multi-dimensional trust metrics to evaluate packet forwarding behavior from sensor nodes. Simulations were conducted to assess the performance of the proposed model, with results indicating improvements in throughput, energy usage, packet delay, and accuracy.

[6] presented a method for modeling and detecting flooding-based Denial-of-Service attacks in wireless ad hoc networks using Bayesian inference. Their work is divided into three sections: 1) Bayesian inference-based mathematical modeling of network SYN traffic, 2) demonstrating the equivalence of Bayesian inference with the exponential weighted moving average, and 3) constructing a Bayesian inference-based system for detecting SYN flooding attacks. A comprehensive assessment, including mathematical modeling and simulation, shows that the proposed strategy effectively prevents various types of flooding-based DoS attacks in wireless ad hoc networks, offering improved detection accuracy and an exceptionally low false detection rate.

[8] introduced two modules for DoS and SPAM detection: a statistics-based DoS detection module and a call behavior-based SPAM detection module. The statistics-based module examines SIP traffic to detect potential DoS attacks. If a DoS attack is detected, SIP packets are discarded, and adaptive thresholds are adjusted for regular traffic. The second module focuses on SPAM detection by analyzing call establishment statistics. SIP packets are categorized by IP, Call-ID, URI, and request mechanism. The collected data is compared to adaptive thresholds at each time interval. If the data falls outside the specified threshold range, an attack is identified.

[17]introduced a SIP parser designed to identify malformed message DoS attacks. The proposed system begins with a lexical analyzer that converts SIP messages into lexemes, stores them in a table, and discards syntactically incorrect messages. Support Vector Machine (SVM) is then used to classify the syntactically correct SIP messages as either normal or malformed. Additionally, kernel tree analysis is employed to save processing time, as it does not require feature space representation. The system achieved a detection accuracy of 99.89%.

According to [18] , machine learning can be employed to analyze large volumes of offline VoIP log files and enhance detection accuracy for low-rate DoS attacks. During the feature extraction phase, they estimated the occurrences of six essential SIP headers within a window of 1000 messages. In the classification phase, five classifiers—neural networks, naive Bayes, random forest, decision trees, and sequential minimal optimization (SMO)—were used to classify 15 distinct DoS and DDoS scenarios. Ad-

ditionally, HMAC anonymization is applied to SIP headers to protect user privacy. The classifiers demonstrated superior performance over Entropy and Hellinger Distance techniques, particularly in terms of false negatives for DDoS scenarios. However, the false positive rates for DDoS scenarios were higher compared to those for DoS scenarios.

Table 1: Comparison of different existing methodologies

Author & Year	Technique	Accuracy (%)	Detection Approach	Key Summary	Limitations
[17]	Machine learning approach- SVM Classifier	99.89	DoS/DDoS	Malware detection is carried out using static, dynamic, and image analysis techniques. A strategy centered on the host based.	Only a tiny dataset of 5000 SIP messages was used. There are no details on DDoS scenarios or classifier fine-tuning.
[19]	IDS based on hybrid feature selection and two-level classifier	96.10	IDS	They suggested a new Intrusion Detection System that utilizes a feature extraction strategy that utilizes evolutionary techniques and a classifier based on Random Forest.	To research a more effective and lightweight technique for detecting detection mistakes and to put our solution into practice.
[20]	Trees ensemble classifiers	97.40	DoS/DDoS	Entropy estimation, clustering, information gain ratio, and Extra-Trees ensemble classifiers are used in a semi-supervised DDoS detection technique.	To assess how well it performs in real-world dataset scenarios
[21]	Particle Swarm Optimization (PSO)-based Fast Learning Network (FLN)	96.5.	0-Day	Utilizing the well-recognized KDD99 dataset, the model was used for the purpose of intrusion detection and subsequently verified	Concentrated on the port's use profile. Attacks with a high volume
[22]	A hybrid approach using Snot IDS	98.2	0-Day	Based on the frequency of exploits, the ranking system provides a likelihood of exploit.	Creates a graph of attacks at certain time stamps Attacks with a high volume

3 Preliminaries

The five preliminary techniques applied to detect DoS attacks in our proposed approach are:

- Hellinger Distance
- Naïve Bayes
- Support Vector Machine
- Logistic Regression
- K Nearest Neighbors

3.1 Hellinger distance

Hellinger Distance is a metric commonly used to measure the similarity (or dissimilarity) between two probability distributions[28]. The value of the Hellinger Distance between any two probability distributions ranges from 0 to 1, where 0 indicates identical distributions and 1 indicates completely divergent distributions. For two discrete probability distributions, P = (p_0, p_1, p_2, …, p_n) and Q = (q_0, q_1, q_2, …, q_n), the Hellinger distance is given as,

$$HD^2(P,Q) = \frac{1}{2}\sum_{i=0}^{n-1}((\sqrt{P_i} - \sqrt{Q_i})^2)^{\frac{1}{2}} \quad (1)$$

which is directly related to the Euclidean norm of the difference of the square root vectors, i.e.

$$HD(P,Q) = \frac{1}{\sqrt{2}}||\sqrt{P} - \sqrt{Q}||_2 \quad (2)$$

Table 2: Comparison of different existing methodologies

Author & Year	Technique	Accuracy (%)	Detection Approach	Key Summary	Limitations
[23]	Bayesian networks -based approach	96.87	0-Day	Graph nodes are made up of file, process, and other instances. It is Host-centric.	The availability of correct evidence is critical to performance.
[24]	GAN based on deep autoencoder	98.45	0-Day	In order to identify zero-day malware, noise is added to it.	Malware detection with a fixed duration
[25]	Decision Tree, Genetic Algorithm	97.77	Signature Based	NSL-KDD , UNSW-NB15 and KDDCup99	On the UNSW-NB15 dataset, performance is poor. Not based on real-time traffic analysis.
[26]	Fuzziness based learning using FF-NN	99.21	Anomaly Based	NSL-KDD	Aims to improve classification accuracy Uses an older dataset. Hasn't been tested on real-time traffic
[27]	A two-step hybrid technique is proposed, which utilizes binary classification and MultiSVM.	97.53	Signature Based	On the CICIDS 2018 dataset, the proposed technique produces credible findings.	SVM performance is dependent on rule correctness Dataset is older Not tested on real-time traffic

3.2 Naïve Bayes

Nave Bayes is a classification technique based on Bayes' theorem and the assumption that the existence of one attribute does not imply the presence of another [29]. In other words, the features are said to be independent. For instance, given a class y and a dependent feature vector [x1, x2, …, xn], we have-

$$P(y|x_1, ..., x_n) = \frac{P(y)P(x_1, ..., x_n)|y)}{P(x_1....., x_n)} \quad (3)$$

And using the assumption that all the xi 's are independent,

$$P(x_i|y, x_i, ..., x_i - 1, x_i + 1, ..., x_n) = P(x_i|y) \quad (4)$$

On simplifying, following result is obtained

$$P(y|x_1, ..., x_n) = (P(y)\prod_{i=1}^{n}\frac{P(y)}{P(x_1 \ldots, x_n)} \quad (5)$$

which is the result of Bayes' theorem.

3.3 Support Vector Machine

Support Vector Machines (SVMs) are a class of machine learning algorithms grounded in statistical learning theory. These techniques are commonly used for classification, regression, and outlier detection [30] In recent years, SVMs have also been extensively applied in pattern recognition. One of the key strengths of SVMs is their ability to minimize empirical classification error while maximizing the geometric margin. This characteristic is why SVMs are often referred to as Maximum Margin Classifiers. The SVM approach is based on Structural Risk Minimization (SRM).

An SVM maps the input vector into a higher-dimensional space where a maximum separating hyperplane is constructed. This hyperplane is flanked by two parallel hyperplanes on either side. The goal is to position the separating hyperplane so that the margin, or distance between the two parallel hyperplanes, is maximized. An illustration of the Support Vector Machine can be seen in Figure 4.

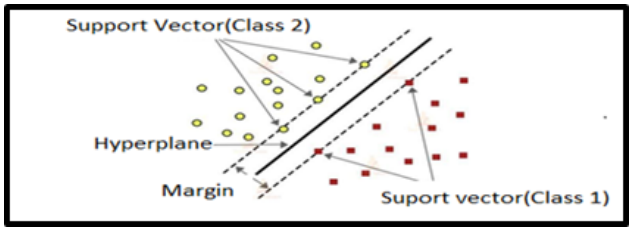


Figure 4: Support Vector Machine illustration

3.4 Logistic Regression

Logistic Regression is a statistical approach for analyzing data, commonly used for classification [31]. Often used for binary classification, it can also be extended for multiclass classification. It includes a logistic function that accepts a real-world input y and returns a number between 0 and 1. Often it is the sigmoid function. For example, if we have a variable y which is linearly dependent on x as

$$y = \beta_0 + \beta_1 x \tag{6}$$

The logistic function may therefore be expressed as follows:

$$\sigma(y) = \frac{1}{(1 + e^{-}\beta_0 + \beta_1 x)} \tag{7}$$

which will scale the values of y between 0 and 1.

3.5 K Nearest Neighbors

K Nearest Neighbors (KNN) is a simple, yet powerful, instance-based machine learning algorithm used for both classification and regression tasks. It operates based on the principle of finding the closest data points in the feature space to make predictions.[32]. Then, majority voting among the neighboring records is used to decide to which class the record belongs to. In other words, the neighbors of the record are used to classify it.The parameter K represents the number of nearest neighbors considered when making a prediction. The choice of K can significantly impact the performance of the algorithm: Small K: The model may be overly sensitive to noise and outliers, potentially leading to overfitting. Large K: The model may smooth out the decision boundary, which can help with generalization but may also blur the distinction between classes. illustration of logistic Regression can be shown in figure 5.

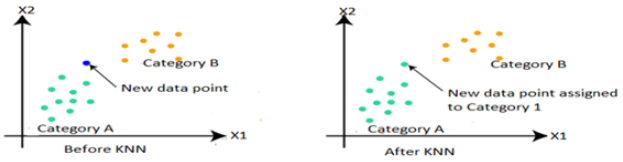


Figure 5: K Nearest Neighbor illustration

4 Proposed detection model

We have proposed an approach for detecting DoS attacks using the Hellinger Distance method. DoS attacks aim to overwhelm server resources, thereby preventing legitimate clients from accessing the services. To better understand the impact of such attacks, we can apply Little's Law, which is a fundamental principle in queuing theory. Little's Law states that the average number of items in a queuing system (L) is equal to the average arrival rate multiplied by the average time an item spends in the system (W) [33]

$$L = \lambda \cdot W \tag{8}$$

Little's Law describes the relationship in a queuing system where L represents the average number of items in the system, λ is the arrival rate of items into the system, and W is the average time each item spends in the system. DoS attacks aim to overload the system queue, thereby preventing legitimate customers from receiving service. By introducing sophisticated computational tasks to the target device, DoS attacks can either increase the packet arrival rate (λ) or extend the per-packet processing time (W).

In this section, we describe a method to detect the proposed DoS attack using a statistical abnormality measurement technique. This detection method is motivated by the observation that various SYN messages have a strong correlation between them.

For instance, every SYN message is typically followed by an ACK message. This balance is disturbed when there is an increased number of DECLINE messages, which often occurs during a DoS attack. We exploit this deviation in message patterns to detect the proposed DoS attacks.

Our detection method operates in two phases: the training phase and the testing phase. In the training phase, we

collect and create a normal behavior profile of SYN operations. In the testing phase, we compare the profile of current SYN operations with the previously generated profile to detect DoS attacks. For this comparison, we opted for methods that compute the distance between two probability distributions. Some of the popular distance metrics used are Bhattacharyya Distance[34], Total Variation Distance [35], Mahalanobis Distance [35], Kullback-Leibler Divergence[36] , and Hellinger Distance [28]. In these equations, μ represents the mean of the corresponding vectors, σ is the standard deviation, and C^{-1} is the covariance matrix generated from training intervals (where each interval is a probability distribution). Here, P and Q are N-dimensional vectors. The proposed model for detecting DoS attacks is comprised of three main phases, as illustrated in Figure 6:

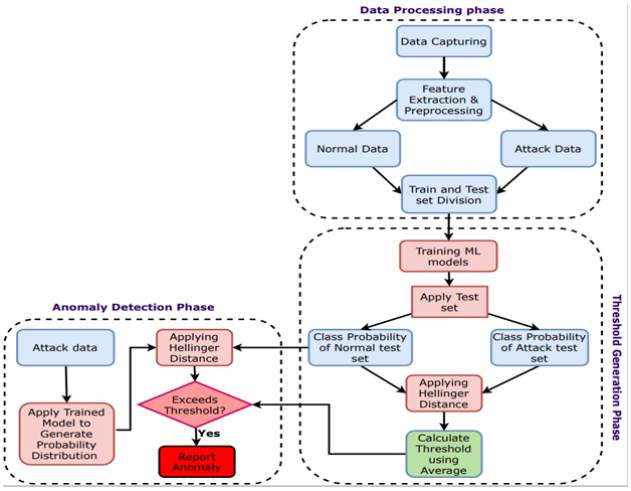


Figure 6: Proposed model for DoS attack detection using Hellinger distance

1. Data Processing Phase
2. Threshold Generation Phase
3. Anomaly Detection Phase.

The following subsections depict the proposed model in detail.

4.1 Data processing phase

In the initial phase of our proposed model, we utilize the Wireshark tool to capture all network traffic data. This step is essential for gathering comprehensive packet data over a span of approximately 20 days. We start by collecting normal traffic data and then proceed to launch a DoS attack on the server, capturing traffic data during the attack.

After gathering the data, we move on to preprocessing, where we normalize the key features of both training and testing datasets to ensure consistency. Next, we conduct feature extraction to identify and select the most relevant features using various extraction techniques.

We then apply different classification methods, such as Logistic Regression and Support Vector Machine, to determine the probabilities for two categories: ”DoS Attack” and ”Normal.” The model is first trained on a subset of the data, and then we use this trained model to compute probabilities for the remaining data batches.

Following this, we calculate the Hellinger Distances for these probabilities. We establish a threshold Hellinger Distance, which is used for comparison with other calculated distances. If a calculated distance is less than or equal to the threshold, it suggests no attack is occurring; if it is greater than the threshold, it indicates a potential attack.

4.1.1 Data capturing

In this subsection of the proposed model, traffic collection methods observe the flow of data remotely and record accessible information for network quality monitoring, traffic estimation, and attack detection and prevention. Traffic can be collected via packets, flows, or logs. The HTTP protocol’s packet. Packets are commonly used in everyday network administration processes for fault rectification, configuration control, performance management, and security monitoring. We gathered two weeks of normal HTTP traffic from the web server set up in our testbed and used this data for training purposes. The time interval size was set to Δ T=10 minutes, resulting in a total of 2016 intervals. Using these intervals, we created a normal HTTP traffic profile.For testing purposes, we generated an additional week of typical traffic using the same setup. We repeated the experiment, incorporating DoS attacks at various rates alongside the regular requests generated by the script. This resulted in two distinct scenarios (one for normal traffic and one for attack instances), both consisting of 2,016 intervals. We also used two traditional type of data set which are NSL KDD Dataset and UNSW_NB15 dataset for comparison purpose.

Various convenient network packet capture techniques are often employed for data gathering and later analysis. TCPdump[37] and Wireshark[38] are two well-known examples. There are two methods for capturing packets: active data collection and passive data collection. Active data collection methods typically inject test data into traffic and wait for replies to assess network quality, while passive data collection methods monitor network traffic using software or hardware monitoring tools. TCPdump and Wireshark are examples of passive, software-based packet capture techniques.

4.1.2 Preprocessing

Data preprocessing is a crucial stage in any approach or process aimed at improving outcomes. Data gathered from various sources is often unstructured, leading to issues such as out-of-range values (e.g., income: -100), impossible data combinations (e.g., Gender: Male, Pregnant: yes), missing information, and more. Analyzing data that has not been

thoroughly checked for such issues can lead to false conclusions. As a result, processing techniques such as Min-Max scaling, one-hot encoding, label encoding, and handling of null values are applied to process datasets. Preprocessing involves steps such as cleaning, normalization, transformation, feature extraction, and selection. Raw data often contains noise, missing values, and inconsistencies, which can affect the outcomes. To improve data quality and, subsequently, the results, raw data is preprocessed to enhance efficiency. Data preprocessing is one of the most important steps in data mining, as it deals with the preparation and modification of the initial dataset. The four types of data preparation procedures are data cleaning, data integration, data transformation, and data reduction.

4.1.3 Feature extraction

Feature extraction is employed when an algorithm has a large amount of input data, much of which may be redundant or irrelevant. It is essential to accurately identify the characteristics that determine whether an input is classified as normal or an attack, as the feature extraction process organizes these features into a more manageable subset of data. This process also reduces the amount of data that needs to be processed, resulting in minimal computational overhead [**35**]. The outcome of the feature extraction step is a vector containing the frequencies of the extracted features. These features are selected to achieve maximum classification accuracy. The time required to extract features from a dataset depends on the methods used. The feature extraction technique directly influences the system's efficiency, resilience, and accuracy.

1. **Chi-squared test**: A statistical hypothesis test, known as the chi-square test, is used to determine whether there is a significant difference between the observed and expected frequencies of one or more categories[39]. The test requires the formulation of both a null and an alternative hypothesis. The null hypothesis asserts that there is no significant difference between the expected and observed frequencies, while the alternative hypothesis suggests that the observed frequencies significantly deviate from the expected ones. The significance level is the threshold at which we can confidently state that this difference is not due to chance. For most scientific experiments, a significance level of 0.05 is commonly considered. Equation 9 presents the definition of the chi-square test statistic.

$$X^2 = \sum_{n=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{9}$$

Where n is the number of categories, and O_i and E_i are the observed and expected numbers of cases in the i^{th} category, respectively. If O_i and E_i for each category are closer to one another, the X^2 value is relatively small. The X^2 value will increase in proportion to the difference between O_i and E_i.

2. **Pearson correlation**: It measures the relationship between two features or variables. The Pearson Correlation between two features is defined in equation 10 as [40]:

$$r = \frac{\sum(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum(x_i - \overline{x})^2 \sum(y_i - \overline{y})^2}} \tag{10}$$

Where:
r = Correlation coefficient
x_i = Values of the x variable in a sample.
$\overline{x}$ = Mean of the values of the x-variable
y_i = Values of the yvariable in a sample.
$\overline{y}$ = Mean of the values of the y-variable

The value of r can range from -1 to 1. If r=0, it indicates no relationship between the variables. r>0 signifies a positive correlation, where an increase in the value of one variable results in an increase in the value of the other. Conversely, r<0 indicates a negative correlation, where an increase in the value of one variable leads to a decrease in the value of the other.

3. **Feature importance using tree-classifier**: In feature selection, tree-based techniques are highly effective [41]. Feature importance is determined by the reduction in node impurity, weighted by the probability of reaching that node. The node probability is calculated by dividing the number of samples reaching the node by the total sample size. The relevance of a feature is directly proportional to its importance value. The feature importance for the UNSW_NB15 dataset and the NSL-KDD dataset is illustrated in Figures 7 and 8, respectively.

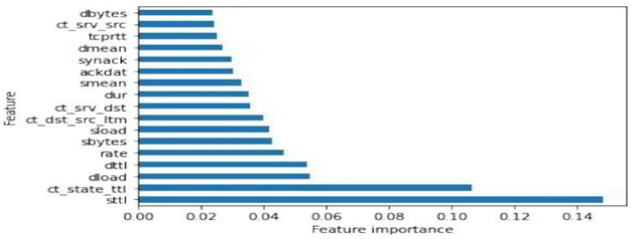


Figure 7: Feature importance of UNSW_NB15 dataset

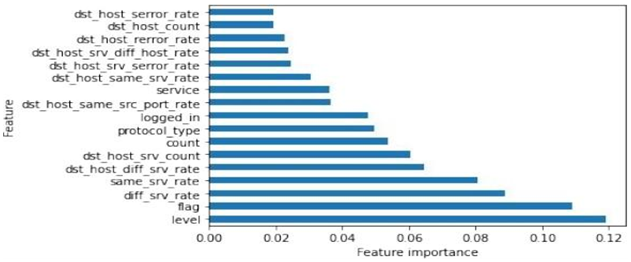


Figure 8: Feature importance of NSL-KDD dataset

4.2 Threshold generation phase

We selected time intervals of size $\Delta T = 10$ minutes to evaluate the detection performance. As previously mentioned, we calculate the Hellinger distance statistic for each time interval using Equation 1, checking for any significant deviations from the expected testing profile. The detection performance of the proposed scheme is assessed using recall and the false positive rate (FPR) as performance indicators. The values for recall and FPR are given by Equations 11 and 12, respectively. In these equations, the variables t_p, t_n, f_p, and f_n represent the number of attack intervals correctly identified as attacks, the number of normal intervals correctly identified as normal, the number of normal intervals incorrectly identified as attacks, and the number of attack intervals incorrectly identified as normal, respectively.

$$Recall = \frac{t_p}{t_p + f_n} \tag{11}$$

$$FPR = \frac{f_p}{f_p + t_n} \tag{12}$$

The detection performance of the proposed scheme is presented in Table 8. These results were obtained at a significance level of 0.05. For a given time interval, the alternative hypothesis (H_A) was accepted if the calculated Hellinger Distance exceeded the threshold value. However, the null hypothesis (H_N) was accepted if the Hellinger Distance for that interval was below the minimum threshold.

4.2.1 Hellinger distance computing

Hellinger distance[42] is a metric that is often used to determine the similarity (or dissimilarity) between two probability distributions. The value of Hellinger Distance between any two probability distributions lies between 0 and 1, where 0 means absolutely equal distributions and 1 means very different or divergent distributions. Hellinger distance is a statistical hypothesis testing technique used to determine if there is a significant deviation between two probability distributions of one or more categories. This test involves proposing two hypotheses—the null hypothesis and the alternative hypothesis. The null hypothesis states that there is no significant difference between the two probabilities, while the alternative hypothesis asserts that the two probabilities deviate from each other for some reason.

The confidence with which we can conclude that this difference is not due to chance is known as the significance level, denoted by α. Typically, a significance level of 0.05 is used in most scientific experiments. The Hellinger distance statistic is defined as shown in Equation 13. For two discrete probability distributions, P = (p_1, p_2, p_3, …, p_n) and = (q_1, q_2, q_3, …, q_n), the Hellinger distance is given as,

$$HD^2(P,Q) = \frac{1}{2}\sum_{i=1}^{n}((\sqrt{P_i}^2 - \sqrt{Q_i}^2)^{\frac{1}{2}} \tag{13}$$

where $\sum_{i=1}^{n} p_i = 1$ and $\sum_{i=1}^{n} q_i = 1$

A Sensitivity Analysis graph can be used to determine the threshold value. If the obtained Hellinger distance value exceeds this threshold, we can reject the null hypothesis H_N. However, if the Hellinger distance value is below the predefined threshold, we do not have enough evidence to reject H_N and accept the alternative hypothesis H_A.

4.3 Anomaly detection phase

Our detection system operates in two key phases: the training phase and the testing phase. During the training phase, we collect data and establish a profile of typical behavior, which is then compared to the profile generated during the testing phase to identify potential DoS attacks. In this study, we employed techniques that measure the distance between two probability distributions. Notable distance metrics, as provided in Equations 14, 15, 16, 17, and 18, include Bhattacharyya Distance[34], Total Variation Distance [35], Mahalanobis Distance [35], Kullback-Leibler Divergence[36] , and Hellinger Distance [28]. In these equations, C^{-1} is the covariance matrix created using training intervals, and μ is the mean of the corresponding vectors and σ is their standard deviation (Each interval is a distribution of probabilities.) P and Q are N-dimensional vectors in this case.

$$d_B(P,Q) = \frac{1}{4}loglog(\frac{1}{4}(\frac{\sigma_p^2}{\sigma_q^2} + \frac{\sigma_q^2}{\sigma_p^2}) + \frac{1}{4}(\frac{(\sigma_p - \sigma_q)^2}{(\sigma_p^2 + \sigma_q^2)}) \tag{14}$$

$$d_T(P,Q) = \frac{1}{2}||P - Q|| \tag{15}$$

$$d_M(P,Q) = \sqrt{((Q - \sigma_p)^t C^-1 (Q - \sigma_p))} \tag{16}$$

$$d_K(P,Q) = \sum_i (P_i loglog(\frac{P_i}{Q_i}) \tag{17}$$

$$HD^2(P,Q) = \frac{1}{2}\sum_{i=0}^{k-1}((\sqrt{P_i} - \sqrt{Q_i})^2)^{\frac{1}{2}} \tag{18}$$

To identify a DoS attack, we selected the Hellinger distance to compare the training and testing probability distributions. In the following section, we provide theoretical justifications for choosing Hellinger Distance over other metrics. Additionally, Part 4.4 presents experimental data demonstrating why Hellinger Distance is the most suitable distance metric for our requirements.

4.3.1 Picking the suitable metric for distance

We used Hellinger Distance to identify DoS attacks due to following theoretical reasons listed below.

1. Lightweight Computation: Comparing two probability distributions using the Hellinger Distance avoids computationally intensive tasks, such as matrix inversion or covariance calculations, which are required for metrics like the Mahalanobis distance. Consequently,

the Hellinger Distance serves as a more efficient alternative for Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) in detecting and mitigating proposed DoS attacks.

2. Natural Lower and Upper Bounds: It is important to note that the value of d_H will always fall between 0 and 1, with 0 denoting perfect similarity and 1 denoting the greatest dissimilarity between P and Q. As a result, Hellinger distance has natural lower and upper bounds of 0 and 1, which are not present in other distance measuring techniques.

3. Yielding Finite Distance Value: Hellinger Distance does not require any such reliance between two probability distributions, in contrast to Kullback-Leibler Divergence, which is only defined if $(Q_i{=}0)P_i{=}0, 1$. In some circumstances, Kullback-Leibler Divergence provides an infinite value, while Hellinger Distance produces an appropriate finite distance value between 0 and 1.

4.4 Experimental reasons to choose hellinger distance as distance metric

We conducted several tests that highlighted the Hellinger Distance as superior to other metrics for measuring the gap between two probability distributions. To detect the proposed DoS attack, we evaluated each of the previously discussed distance metrics. Table 2 outlines the various methods used to measure distances between two probability distributions. From these evaluations, we ruled out the Mahalanobis Distance and the Kullback-Leibler Divergence—the former lacked a natural bound, and the latter produced an indeterminate distance if the probability of one or more events during the testing period dropped to zero. After excluding these metrics, we tested 18 normal intervals and identified the maximum distance value among them. We then chose the method that showed the greatest difference between the maximum and minimum distances measured during normal intervals and the distance measured during the proposed attack's interval. For example, as shown in Table 3, the maximum distance measured using the Bhattacharyya Distance during normal intervals was 0.002132, while the distance measured during the proposed attack's interval was 0.0936, resulting in a difference of 0.091468. Similarly, the Total Variation Distance and Hellinger Distance showed differences of 0.129128 and 0.280757, respectively. Since the Hellinger Distance demonstrated the largest difference between distances measured during normal and attack intervals, we selected it as the most suitable distance metric. This selection criterion assists network administrators in easily defining a threshold distance and reduces the likelihood of false positives.

Table 3: Distance measured between two probability distributions using different methods

Method	Minimum distance (Normal)	Maximum distance (Normal)	Distance measured (Attack)
Bhattacharyya Distance	0.0000197	0.002132	0.0936
Total Variation Distance	0.00615	0.059072	0.1882
Mahalanobis Distance	0.2432	0.5922	1.9912
Kullback-Leibler Divergence	0.0000758	0.008459	0.1872
Hellinger Distance	0.004381	0.046143	0.3269

4.4.1 Adapting Hellinger distance for DoS attack detection

The described detection system has the following two components

1. Probabilistic Distribution of Training Data: To develop a profile of normal traffic, we observe and generate a distribution profile over a series of n observation intervals, each lasting ΔT. The profile, created during the training phase, includes ten attributes: src_ip, dst_ip, protocol, length, src_port, dst_port, seq_no, len, win_size, and flags. These attributes reflect the probability of occurrence of each type of traffic. To calculate the probability P_i for each attribute, we use Equation 6, where N_i represents the total number of occurrences of attribute i during the training period (nΔT), and N_{total} is the total number of occurrences of all ten attributes over the same period.

$$P_i = N_i / N_{total} \tag{19}$$

2. Probabilistic Distribution of Testing Data: The system is used to identify DoS attacks starting from the $(n+1)^{th}$ period of duration ΔT, once it has been trained and P has been generated. Equation 6 is used to produce a probability distribution Q at every period of duration ΔT, once it has been trained and P has been generated. Equation 6 is used to produce a probability distribution Q at every Δ T-interval. In this instance, N_i stands for the count of events of type i in the interval under examination, and N_{total} for the sum of

events of all 10 categories in the same interval. Q produced in this manner is contrasted with P using the Hellinger distance. A DoS attack is detected if the gap between the two distributions is greater than a threshold.

5 Dataset

We have used 2 types of datasets:

1. Traditional dataset
2. Real time dataset

5.1 Traditional dataset

Among available traditional datasets, we have used 2 of its types [43]:

5.1.1 NSL KDD dataset

KDD 99' Dataset is the predecessor of NSL-KDD Dataset. It serves as an effective Benchmark dataset to help scholars compare among different types of intrusion systems [44].

Table 4: Description of NSL KDD Dataset

S.No.	Name of File	Number of Features	Number of Samples	Labels
1.	KDD Train+ .csv file	43	125972	5 [DoS, U2R, R2L, Probing , Normal]
2.	KDD Test+ .csv file	43	22542	5 [DoS, U2R, R2L, Probing, Normal]

5.1.2 UNSW NB15 dataset

It is one of the most used datasets to examine intrusion detection systems. It is developed in the Cyber Range Lab of the Australian Center for Cybersecurity [45].

Table 5: Description of UNSW_NB15 Dataset

S.No.	Name of File	Number of Features	Number of Samples	Labels
1.	UNSW_NB15 _training.csv	43	82332	10 [DoS, Fuzzers, Analysis, Backdoor, Normal etc.]
2.	UNSW_NB15 _testing.csv	43	175341	10 [DoS, Fuzzers, Analysis, Backdoor, Normal etc.]

5.2 Real time dataset

We collected the real-time dataset, which is divided into the following subsections: network topology used, packet sniffing tool employed, and simulation of the DoS attack.

5.2.1 Network topology used for data collection

As an initial configuration, a server with five nodes was employed: three nodes were responsible for generating normal traffic, while the remaining two were designated for attack traffic generation. One system functioned as the server, equipped with a firewall and the Wireshark packet-sniffing tool to analyze incoming and outgoing traffic. Figure 9 illustrates the network topology used for data collection in our system.

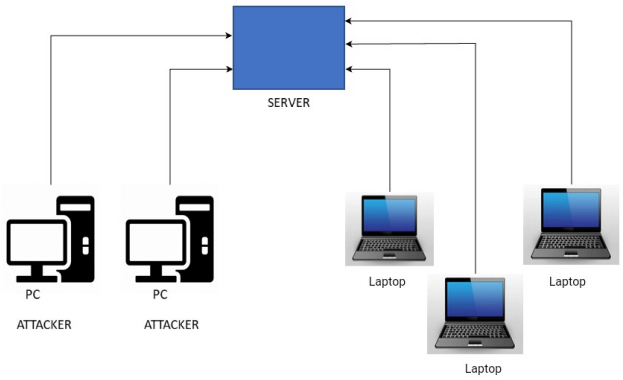


Figure 9: Network topology for data collection

5.2.2 Packet sniffing tool used for data collection

Wireshark was utilized as the packet-sniffing tool for data gathering. It is a free, open-source, and user-friendly packet analyzer with features such as filters and color-coding to differentiate between various types of packets. The data captured by Wireshark can easily be exported to a CSV file. Figure 10 shows a snapshot of the Wireshark tool capturing traffic data during the collection process.

5.2.3 Data collection description

Packet capturing can be accomplished through various methods, with active and passive data collection techniques being the most common. Active data gathering techniques typically inject test data into network traffic and wait for a response, while passive methods monitor traffic using software or hardware tools. Wireshark, a software-based packet capture tool, was used as a passive technique in our approach. During this phase, we employed Wireshark on the server and Kali Linux software to record all traffic data. For our proposed model, data was collected over a span of three weeks. Normal traffic was gathered during the first two weeks, followed by a DoS attack on the server, during which we collected traffic data for the final week. The

description of the real-time dataset and its attributes are presented in Table 4 and Table 5, respectively. A snapshot of the collected dataset is shown in Figure 11.

5.2.4 Simulation of DoS attack

We used the Hping program to send modified packets. This tool allowed us to control the size, quantity, and fragmentation of packets in order to overwhelm the target and bypass the firewall. Command Used : sudo hping3 –rand -source target-ip-address -S -p 80 –flood

Where :–rand-source : It is used to hide attacker's ip address (IP spoofing)

-S : it specifies SYN packets

-p 80 : it specifies an attack is to be made against port 80

-–flood : repeatedly send the packets to destination

Platform used : Kali-Linux (Ubuntu), Oracle VM Virtual Box

Table 6: Description of Real Time Dataset

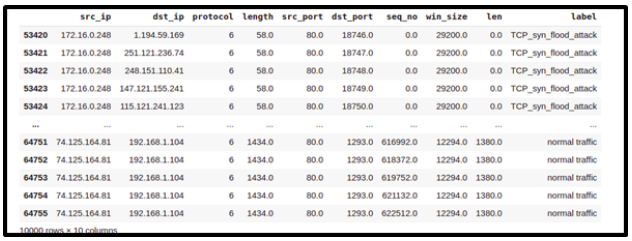
S.No.	Name of file	Number of features	Number of samples	Labels
1.	Real time dataset	10	1241759	2 [Attack, Normal]

Table 7: Candidate Attributes of Real time Dataset with description

Attributes	Description	Type
src_ip	IP address of source	Numeric
dst_ip	IP address of destination	Numeric
protocol	The network layer protocols	Numeric
length	Length of packet	Numeric
src_port	Port number of the source	Numeric
dst_port	Port number of the destination	Numeric
seq_no	Sequence number of the packet	Numeric
Win_size	Window size	Numeric
len	Length of TCP packet	Numeric
label	Shows whether the data is attack or normal	Character

6 Result analysis and performance evaluation

In this section, we discuss the tests conducted to evaluate the performance of the proposed detection scheme. The following subsections cover the generation of traffic for training and testing, the experimental evaluation of DoS attack detection, detection performance, and sensitivity analysis of the proposed technique at various significance levels and time interval sizes. The result analysis and performance section will present the overall effectiveness of our model based on performance metrics from four machine learning algorithms using the Hellinger distance technique. We implemented our model on a custom dataset, achieving excellent results in all three cases of analysis. The outcomes of these cases are discussed below:



	src_ip	dst_ip	protocol	length	src_port	dst_port	seq_no	win_size	len	label
53420	172.16.0.248	1.194.59.169	6	58.0	80.0	18746.0	0.0	29200.0	0.0	TCP_syn_flood_attack
53421	172.16.0.248	251.121.236.74	6	58.0	80.0	18747.0	0.0	29200.0	0.0	TCP_syn_flood_attack
53422	172.16.0.248	248.151.110.41	6	58.0	80.0	18748.0	0.0	29200.0	0.0	TCP_syn_flood_attack
53423	172.16.0.248	147.121.155.241	6	58.0	80.0	18749.0	0.0	29200.0	0.0	TCP_syn_flood_attack
53424	172.16.0.248	115.121.241.123	6	58.0	80.0	18750.0	0.0	29200.0	0.0	TCP_syn_flood_attack
...	...	...	...	...	...	...	...	...	...	...
64751	74.125.164.81	192.168.1.104	6	1434.0	80.0	1293.0	616992.0	12294.0	1380.0	normal traffic
64752	74.125.164.81	192.168.1.104	6	1434.0	80.0	1293.0	618372.0	12294.0	1380.0	normal traffic
64753	74.125.164.81	192.168.1.104	6	1434.0	80.0	1293.0	619752.0	12294.0	1380.0	normal traffic
64754	74.125.164.81	192.168.1.104	6	1434.0	80.0	1293.0	621132.0	12294.0	1380.0	normal traffic
64755	74.125.164.81	192.168.1.104	6	1434.0	80.0	1293.0	622512.0	12294.0	1380.0	normal traffic

10000 rows × 10 columns

Figure 10: A batch of dataset

6.1 Architecture of testbed

To assess the detection performance of the proposed technique, we set up a testbed similar to the one illustrated in Figure 8. We designated one computer as a web server, connecting it to the internet. This server ran the Ubuntu 16.04 LTS operating system and was configured with Apache 2.4.23 software to handle HTTP requests. Apache was chosen for our testbed due to its status as the most widely used web server software globally [35]. The web server was equipped with an Intel Core 2 Duo Processor and 4GB of physical memory. We hosted a sample website on this server, consisting of 25 web pages that provided brief tutorials for an online course. One of these web pages included an image upload field, allowing users to upload an image of 617 Kilobytes to the server. This page also outlined certain conditions (such as image size and format) that needed to be met for successful uploading. Another computer was designated as a traffic generator, simulating the behaviour of web users and sending legitimate HTTP traffic to the web server. This computer ran the Linux Mint operating system, was powered by an AMD Athlon X2 270 Dual-core processor, and had 4GB of physical memory. The behaviour of web users was simulated using a Python program. We also designated another computer as the malicious client, responsible for generating anomalous traffic during different intervals of the testing phase. This computer was equipped with 4GB of physical memory, a dual-core processor, and ran the Ubuntu 16.04 LTS operating system. Both the malicious client and the traffic generator were connected to the internet as well.

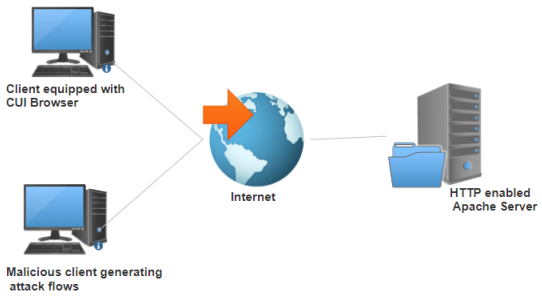


Figure 11: Architecture of testbed

6.2 Traffic generation for training

We collected 14 days of typical HTTP traffic from the web server set up in our testbed and used it for training. A time interval size of $\Delta T = 10$ minutes was used, resulting in a total of 2,016 intervals. These intervals were then used to develop the normal traffic profile.

6.3 Traffic generation for testing

For testing purposes, we generated an additional week of typical traffic using the same setup. We repeated the experiment, incorporating DoS attacks at various rates alongside the regular requests generated by the script. This resulted in two distinct scenarios (one for normal traffic and one for attack instances), both consisting of 2,016 intervals.

6.4 Experimental evaluation of DoS attack detection

We created the distribution profile P using traffic from the first two week as training data for detection purposes. For our experiments, we selected a time interval of 10 minutes. The two weeks of training data generated a total of 2,016 such intervals. We estimated the probability distribution for each of these interval datasets. For testing purposes, we generated an additional week's worth of normal traffic using the same setup. Like the training phase, we used intervals of ΔT=10 minutes during the testing phase. This resulted in 1,008 intervals over the course of one week's worth of normal HTTPS traffic. As previously mentioned, we evaluated the distance between the normal and testing interval probability distributions. From the third day's data, 18 such intervals met this requirement. Table 6 presents the estimated Hellinger distances and the clock times of these 18 intervals. We launched the proposed DoS attack between 3:00 and 3:10 p.m. and 3:10 and 3:20 p.m. Additionally, Table 6 highlights in red the Hellinger distances calculated for these intervals compared to the training profile. It is clear that the Hellinger distances estimated during DoS attack scenarios are significantly higher than those during normal traffic.

Table 8: Detection of normal and DoS attack scenarios

Scenario	Interval	Hellinger Distance	Detection Result	Detection Rate
Normal	5:00pm-5:10pm	0.0045	Normal	100%
	5:10pm-5:20pm	0.0037	Normal	100%
	5:20pm-5:30pm	0.0043	Normal	100%
	5:30pm-5:40pm	0.0027	Normal	100%
	5:40pm-5:50pm	0.0055	Normal	100%
	5:50pm-6:00pm	0.0063	Normal	100%
	10:00am-10:10am	0.0021	Normal	100%
	10:10am-10:20am	0.0230	Normal	100%
	10:20am-10:30am	0.0410	Normal	100%
	10:30am-10:40am	0.0077	Normal	100%
	10:40am-10:50am	0.0557	Normal	100%
	10:50am-11:00am	0.1022	Normal	100%
	2:00pm-2:10pm	0.0200	Normal	100%
	2:10pm-2:20pm	0.0083	Normal	100%
	2:20pm-2:30pm	0.0333	Normal	100%
	2:30pm-2:40pm	0.0099	Normal	100%
	2:40pm-2:50pm	0.0233	Normal	100%
	2:50pm-3:00pm	0.0081	Normal	100%
DoS Attack	3:00pm-3:10pm	0.7623	Anomaly	100%
	3:10pm-3:20pm	0.6772	Anomaly	100%

6.5 Sensitivity analysis

The effectiveness of the proposed approach in detecting objects depends on the threshold set for the Hellinger distance value and the time window used for monitoring various parameters. To minimize false positives and maximize true detections, selecting the appropriate Hellinger distance threshold value and the time interval size ΔT is both crucial and imperative. In this section, we explore the impact of adjusting the time interval ΔT size and the threshold Hellinger distance value on the detection accuracy of the proposed scheme.

To investigate the sensitivity of the scheme to these two parameters, we conducted an experiment by varying the threshold Hellinger distance value and the time interval size ΔT. For this experiment, we generated an additional two weeks of traffic by injecting DoS attacks at a rate of one flow or attack every 300 seconds.

6.5.1 Performance v/s window size

In the first case, we have kept the constant threshold value while varying the window size. We took two values at seven different significance levels a α = 0.005, 0.01, 0.025, 0.05, 0.1, 0.25 and 0.50 then varied the time intervals in steps of 5 minutes, ranging from 5 to 25 minutes. Recall rate increases with increase in both Δ T size and significance level. We observed that regardless of α value, 100% Recall rates were attained for Δ T= 25 minutes, whereas the poorest recall rates were attained for Δ T = 5 minutes and α = 0.005.

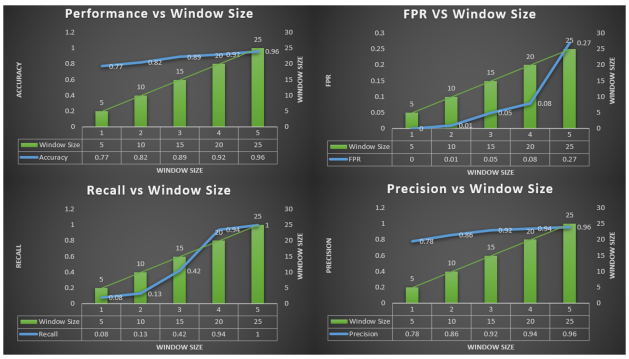


Figure 12: Performance vs window size

6.5.2 Performance v/s threshold

In the second case we have kept the constant window size while varying the threshold value. We varied the threshold value, ranging from 0.45 to 0.75. Recall rate increases with increase in both Δ T size. We observed that in our tests, the worst FPR was found for Δ T = 25 minutes and α = 0.5.

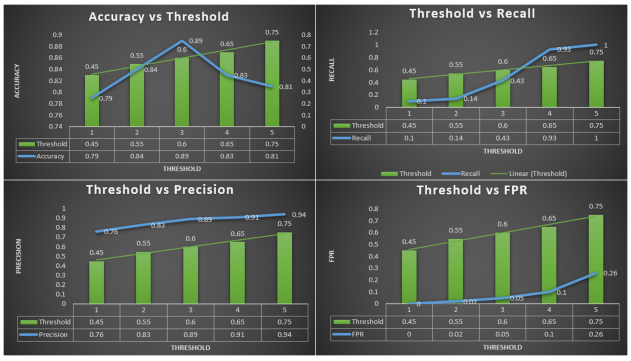


Figure 13: Performance vs threshold

This analysis allows us to determine the optimal time interval size and threshold significance level. A smaller time interval can negatively impact the recall rate (especially at lower significance levels), despite resulting in a very low false positive rate (FPR). Conversely, a larger time interval improves the recall rate but tends to increase the FPR (especially at higher significance levels), which is undesirable. Therefore, choosing the appropriate time interval size and Hellinger distance threshold is critical for achieving optimal detection accuracy with the proposed scheme.

6.6 Detection performance

We evaluate the detection performance of the proposed scheme using metrics such as precision, recall, accuracy, and F1 score. These metrics are computed based on the values obtained from the Hellinger distance threshold and the specified window time. To optimize the scheme's ability to identify objects while minimizing false detections and maximizing true detections, it is crucial to select the appropriate Hellinger distance threshold and time interval size (T). This section explores the detection accuracy of the proposed approach by fixing the threshold and varying the batch size. We conducted tests using three different batch sizes to assess the performance of various algorithms. A threshold value of 0.60 was chosen for all three cases, as our sensitivity analysis indicated that this value yields the highest accuracy.

6.6.1 Performance evaluation metrics

To evaluate the models used in the experiment, as performance metrics, we have chosen accuracy, precision, recall, and F1-score, where TP stands for True Positive, TN for True Negative, FP for False Positive, and FN for False Negative [46]. In Algorithm 1, we have also demonstrated the performance evaluation metrics computation for each of the four ML algorithms.

– **Accuracy:** It is referred to as the number of right predictions divided by the total number of data instances.

$$Accuracy = \frac{(TP + TN)}{(TP + FN + TN + FP)} \tag{20}$$

– **Precision:** The ratio of accurately anticipated positive data to the total quantity of positive data expected is what it's called.

$$Precision = \frac{(TP)}{(TP + FP)} \tag{21}$$

– **Recall:** It is the proportion of accurately predicted positive data in the class to the total amount of data in the class.

$$Recall = \frac{(TP)}{(TP + FN)} \tag{22}$$

– **F1- score:** It is referred to as the harmonic mean of Precision and Recall.

$$F1 - score = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)} \tag{23}$$

– **Computational efficiency:** Computational Efficiency in the context of intrusion detection systems (IDS) and anomaly detection can be evaluated based on various factors, including the time complexity, space complexity, and the overall resource requirements of the detection methods. Here's a detailed examination of computational efficiency concerning different intrusion detection approaches, including our Hellinger Distance-based method:

  – Time Complexity: The Hellinger Distance calculation involves simple mathematical operations like summation and square root, making it computationally efficient. The time complexity is typically linear concerning the number of features or dimensions in the dataset.

  – Space Complexity: Low, as it requires minimal storage for probability distributions and calculated distances.

  – Efficiency: The Hellinger Distance method is computationally efficient compared to more complex statistical or machine learning methods. It avoids intensive operations like matrix inversions and can be implemented with relatively low computational overhead, making it suitable for real-time applications.

6.6.2 Case Study[I]: Batch size = 10000

In this section, we discuss the results when the batch size of our dataset is set to 10,000. We evaluated four machine learning algorithms: Logistic Regression, Naïve Bayes, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN). For each algorithm, we created a graph depicting the relationship between the Hellinger distance and the batch size. Figure 12 illustrates the outcomes, with accompanying charts for each model. Performance metrics, including Precision, Recall, Accuracy, and F1 Score for all four algorithms, are summarized in Table 6.

The Hellinger distance values range between 0 and 1, with a threshold value indicating anomalies if the distance exceeds the threshold, and normalcy if it falls below. This relationship is demonstrated in Fig. 13, which plots Batch Number against Hellinger distance. The graph shows fixed threshold values for each algorithm, with values above the threshold indicating anomalies and those below considered normal.

The accuracy of the Logistic Regression, Naïve Bayes, SVM, and KNN algorithms in this scenario is reported as 83.40%, 90.45%, 87.60%, and 84.70%, respectively.

6.6.3 Case Study[II]: Batch size = 5000

In this section, we present the results for when the batch size of our dataset is 5,000. We used the same four machine learning algorithms and plotted the relationship between the Hellinger distance and batch size for each algorithm. Figure 14 shows the results with corresponding plots for the different models. Performance metrics, including Precision, Recall, Accuracy, and F1 Score for all four algorithms, are summarized in Table 10.

Table 9: Results obtained by different algorithms when batch size 10000

Method	Precision	Recall	Accuracy	F1-score
Logistic Reg	0.871	0.607	0.834	0.715
Naïve Bayes	0.980	0.893	0.904	0.934
SVM	0.921	0.625	0.876	0.744
KNN	0.942	0.589	0.847	0.725

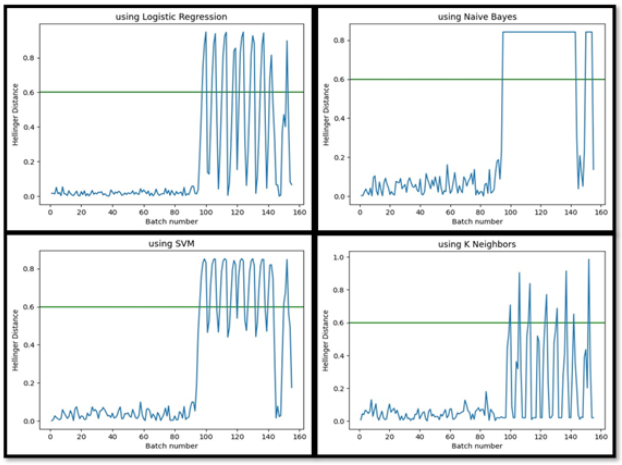


Figure 14: Anomaly detection graph when batch size 10000

Similar to previous analyses, Figure 14 illustrates that the Hellinger distance values range between 0 and 1. A threshold value based on the Hellinger distance indicates anomalies if the distance exceeds this threshold, and normalcy if it is below. Each algorithm maintains a constant threshold value, as clearly visible in the graph.

According to Table 7, the accuracy for Logistic Regression, Naïve Bayes, SVM, and KNN in this case is 88.10%, 91.70%, 86.60%, and 84.00%, respectively.

Table 10: Results obtained by different algorithms when batch size 5000

Algorithm	Precision	Recall	Accuracy	F1-score
Logistic Reg	0.941	0.714	0.881	0.812
Naïve Bayes	0.981	0.928	0.917	0.959
SVM	0.917	0.598	0.866	0.724
KNN	0.9375	0.5357	0.840	0.681

6.6.4 Case Study[III]: Batch size = 20000

In this section, we discuss the results for when the batch size of our dataset is 20,000. We employed the same four machine learning algorithms and created graphs showing the relationship between the Hellinger distance and batch size for each algorithm. Figure 15 presents these results

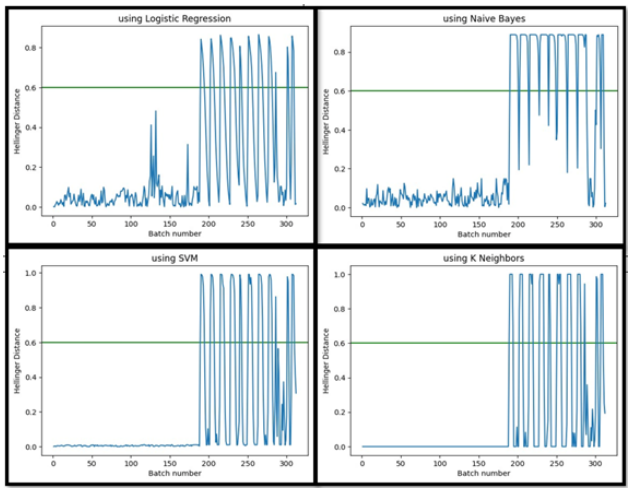


Figure 15: Anomaly detection graph when batch size 5000

with corresponding plots for the different models. Performance metrics, including Precision, Recall, Accuracy, and F1 Score for all four algorithms, are summarized in Table 11.

Figure 15 demonstrates that the Hellinger distance values range from 0 to 1, with a threshold value used to identify anomalies. If the Hellinger distance exceeds this threshold, it is classified as an anomaly; otherwise, it is considered normal.

According to Table 8, the accuracy of Logistic Regression, Naïve Bayes, SVM, and KNN in this case is 85.11%, 92.10%, 89.61%, and 87.01%, respectively.

Table 11: Results obtained by different algorithms when batch size 20000

Algorithm	Precision	Recall	Accuracy	F1-score
Logistic Reg	0.8361	0.6896	0.8511	0.7540
Naïve Bayes	0.9642	0.9310	0.9210	0.9473
SVM	0.9565	0.7586	0.8961	0.8461
KNN	0.9523	0.6896	0.8701	0.7999

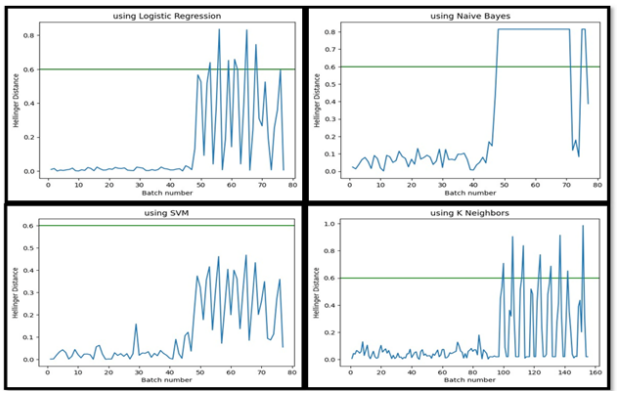


Figure 16: Anomaly detection graph when batch size 20000

6.7 Performance comparison of our own collected real time dataset with existing dataset

In this section, we compare the results obtained from our real-time dataset with those from the NSL-KDD and UNSW_NB15 datasets. The analysis includes all four machine learning algorithms across four performance metrics: Precision, Recall, Accuracy, and F1 Score. The results are summarized in Table 12.

The analysis reveals that the accuracy of all four algorithms on our own real-time dataset is notably higher compared to the NSL-KDD and UNSW_NB15 datasets. Specifically, we achieved a maximum accuracy of 99.41% with our real-time dataset, whereas the NSL-KDD dataset and UNSW_NB15 dataset yielded accuracies of 85.29% and 81.42%, respectively.

Table 12: Result comparison of our own real time dataset with NSL-KDD & UNSW_NB15 dataset

Different Dataset	Algorithm	Precision	Recall	Accuracy	F1-score
Our own Real Time Dataset	Logistic Reg	0.8827	0.6702	0.8553	0.7603
	Naïve Bayes	0.9750	0.9173	0.9941	0.9467
	SVM	0.9315	0.6605	0.8793	0.7714
	KNN	0.9439	0.6047	0.8524	0.7353
NSL-KDD Dataset	Logistic Reg	0.9365	0.7373	0.7957	0.7343
	Naïve Bayes	0.9717	0.9232	0.8371	0.8392
	SVM	0.9226	0.8754	0.8529	0.8734
	KNN	0.9450	0.7121	0.8447	0.8811
UNSW_NB15 Dataset	Logistic Reg	0.9625	0.7411	0.7722	0.7523
	Naïve Bayes	0.9829	0.9169	0.7831	0.8121
	SVM	0.9819	0.8454	0.8021	0.8347
	KNN	0.9902	0.6921	0.8142	0.8551

6.8 Comparison with prior approaches

We compared the performance of our proposed approach for DoS attack detection with existing approaches. Our approach yielded better results than the prior work, as illustrated in Figure 17.

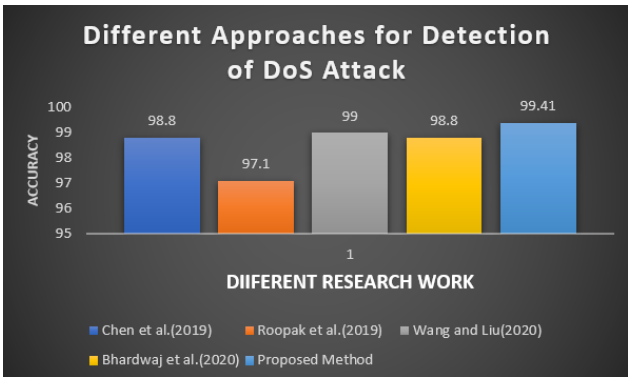


Figure 17: Comparative analysis of proposed approach with existing approaches

7 Conclusion and future work

In this paper, we propose a novel DoS attack detection method using the Hellinger Distance technique. This approach introduces a robust and unique model for identifying DoS attacks. The proposed DoS Attack Detection Model is organized into three phases: (a) Data Processing, (b) Threshold Generation, and (c) Anomaly Detection. Data is collected within a virtual environment, and the results are analyzed for binary classification.

The model demonstrates a binary classification accuracy of 99.41% on real-time attack data. In comparison, the NSL-KDD benchmark dataset and the UNSW_NB15 dataset show binary classification accuracy's of 85.29% and 81.42%, respectively. We identified DoS attacks by setting a threshold value and analyzed its impact using various machine learning models in conjunction with the Hellinger distance.

Several case studies were conducted, each corresponding to different batch sizes and Hellinger distances. The models were evaluated using precision, recall, accuracy, and F1 score, yielding promising results. The performance of all four algorithms was assessed across these metrics.

Looking ahead, the Hellinger distance could be used to detect deviations in the normal behavior of different protocols to identify network anomalies. The proposed approach has potential for adaptation to detect attacks against IoT protocols such as MQTT, AMQP, and CoAP. Future work will focus on exploring the application of this approach to these protocols.

References

[1] Z. Tan, A. Jamdagni, X. He, P. Nanda, R. P. Liu, and J. Hu, "Detection of denial-of-service attacks based on computer vision techniques," *IEEE transactions on computers*, vol. 64, no. 9, pp. 2519–2533, 2014.

[2] J. Pougajendy and A. R. K. Parthiban, "Detection of sip-based denial of service attack using dual cost formulation of support vector machine," *The Computer Journal*, vol. 60, no. 12, pp. 1770–1784, 2017.

[3] S. Wankhede and D. Kshirsagar, "Dos attack detection using machine learning and neural network," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, IEEE, 2018, pp. 1–5.

[4] M. Agarwal, D. Pasumarthi, S. Biswas, and S. Nandi, "Machine learning approach for detection of flooding dos attacks in 802.11 networks and attacker localization," *International Journal of Machine Learning and Cybernetics*, vol. 7, pp. 1035–1051, 2016.

[5] N. Tripathi, N. Hubballi, and Y. Singh, "How secure are web servers? an empirical study of slow http dos attacks and detection," in *2016 11th International Conference on Availability, Reliability and Security (ARES)*, IEEE, 2016, pp. 454–463.

[6] N. Nishanth and A. Mujeeb, "Modeling and detection of flooding-based denial-of-service attack in wireless ad hoc network using bayesian inference," *IEEE Systems Journal*, vol. 15, no. 1, pp. 17–26, 2020.

[7] C. Anand and N. Vasuki, "Trust based dos attack detection in wireless sensor networks for reliable data transmission," *Wireless Personal Communications*, vol. 121, no. 4, pp. 2911–2926, 2021.

[8] J. Lee, K. Cho, C. Lee, and S. Kim, "Voip-aware network attack detection based on statistics and behavior of sip traffic," *Peer-to-Peer Networking and Applications*, vol. 8, pp. 872–880, 2015.

[9] P. Alcoy, S. Bjarnason, P. Bowen, C. Chui, K. Kasavchenko, and G. Sockrider, "Netscout arbor's 13th annual worldwide infrastructure security report," *Netscout Systems, Inc., Burlington, MA, USA, Tech. Rep*, 2018.

[10] J. Pescatore, "Ddos attacks advancing and enduring: A sans survey," *Tech. Rep.*, 2014.

[11] V. Durcekova, L. Schwartz, and N. Shahmehri, "Sophisticated denial of service attacks aimed at application layer," in *2012 ELEKTRO*, 2012, pp. 55–60. DOI: 10.1109/ELEKTRO.2012.6225571.

[12] Z. A. Baig, S. Sanguanpong, S. N. Firdous, T. G. Nguyen, C. So-In, *et al.*, "Averaged dependence estimators for dos attack detection in iot networks," *Future Generation Computer Systems*, vol. 102, pp. 198–209, 2020.

[13] S. Sharma, Y. Gigras, R. Chhikara, and A. Dhull, "Analysis of nsl kdd dataset using classification algorithms for intrusion detection system," *Recent Patents on Engineering*, vol. 13, no. 2, pp. 142–147, 2019.

[14] S. Ramesh, C. Yaashuwanth, K. Prathibanandhi, A. R. Basha, and T. Jayasankar, "An optimized deep neural network based dos attack detection in wireless video sensor network," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–14, 2021.

[15] Z. He, T. Zhang, and R. B. Lee, "Machine learning based ddos attack detection from source side in cloud," in *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, IEEE, 2017, pp. 114–120.

[16] N. Gao, L. Gao, Q. Gao, and H. Wang, "An intrusion detection model based on deep belief networks," in *2014 Second international conference on advanced cloud and big data*, IEEE, 2014, pp. 247–252.

[17] A. Akbar, S. M. Basha, S. A. Sattar, and S. Raziuddin, "An intelligent sip message parser for detecting and mitigating ddos attacks," *Int. J. Innov. Eng. Technol*, vol. 7, no. 2, pp. 1–7, 2016.

[18] Z. Tsiatsikas, A. Fakis, D. Papamartzivanos, D. Geneiatakis, G. Kambourakis, and C. Kolias, "Battling against ddos in sip: Is machine learning-based detection an effective weapon?" In *2015 12th International Joint Conference on e-Business and Telecommunications (ICETE)*, IEEE, vol. 4, 2015, pp. 301–308.

[19] Z. Liu and Y. Shi, "A hybrid ids using ga-based feature selection method and random forest," *International Journal of Machine Learning and Computing*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:247829090.

[20] M. Zekri, S. El Kafhali, N. Aboutabit, and Y. Saadi, "Ddos attack detection using machine learning techniques in cloud computing environments," in *2017 3rd international conference of cloud computing technologies and applications (CloudTech)*, IEEE, 2017, pp. 1–7.

[21] P. J. Sajith and G. Nagarajan, "Intrusion detection system using deep belief network & particle swarm optimization," *Wirel. Pers. Commun.*, vol. 125, no. 2, pp. 1385–1403, Jul. 2022, ISSN: 0929-6212. DOI: 10.1007/s11277-022-09609-x. [Online]. Available: https://doi.org/10.1007/s11277-022-09609-x.

[22] U. K. Singh, C. Joshi, and D. Kanellopoulos, "A framework for zero-day vulnerabilities detection and prioritization," *Journal of Information Security and Applications*, vol. 46, pp. 164–172, 2019.

[23] X. Sun, J. Dai, P. Liu, A. Singhal, and J. Yen, "Using bayesian networks for probabilistic identification of zero-day attack paths," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 10, pp. 2506–2521, 2018.

[24] J.-Y. Kim, S.-J. Bu, and S.-B. Cho, "Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders," *Information Sciences*, vol. 460, pp. 83–102, 2018.

[25] D. Papamartzivanos, F. G. Mármol, and G. Kambourakis, "Dendron: Genetic trees driven rule induction for network intrusion detection systems," *Future Generation Computer Systems*, vol. 79, pp. 558–574, 2018.

[26] R. A. R. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," *Information sciences*, vol. 378, pp. 484–497, 2017.

[27] A. V. Turukmane and R. Devendiran, "M-multisvm: An efficient feature selection assisted network intrusion detection system using machine learning," *Comput. Secur.*, vol. 137, p. 103 587, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:265137226.

[28] X. Cao, *Model selection based on expected squared Hellinger distance*. Colorado State University, 2007.

[29] B. Sharmila and R. Nagapadma, "Intrusion detection system using naive bayes algorithm," in *2019 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*, IEEE, 2019, pp. 1–4.

[30] S. Krishnaveni, P. Vigneshwar, S. Kishore, B. Jothi, and S. Sivamohan, "Anomaly-based intrusion detection system using support vector machine," in *Artificial intelligence and evolutionary computations in engineering systems*, Springer, 2020, pp. 723–731.

[31] E. Besharati, M. Naderan, and E. Namjoo, "Lr-hids: Logistic regression host-based intrusion detection system for cloud environments," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, pp. 3669–3692, 2019.

[32] R. Wazirali, "An improved intrusion detection system based on knn hyperparameter tuning and cross-validation," *Arabian Journal for Science and Engineering*, vol. 45, no. 12, pp. 10 859–10 873, 2020.

[33] J. D. Little and S. C. Graves, "Little's law," *Building intuition: insights from basic operations management models and principles*, pp. 81–100, 2008.

[34] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distribution," *Bulletin of the Calcutta Mathematical Society*, vol. 35, pp. 99–110, 1943.

[35] N. Hubballi and N. Tripathi, "A closer look into dhcp starvation attack in wireless networks," *Computers & Security*, vol. 65, pp. 387–404, 2017.

[36] N. Noor Imanina and H. IGARASHI, "Policy gradient method using fuzzy controller in policies and its application,"

[37] J. Therdphapiyanak and K. Piromsopa, "An analysis of suitable parameters for efficiently applying k-means clustering to large tcpdump data set using hadoop framework," in *2013 10th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, 2013, pp. 1–6. DOI: 10.1109/ECTICon.2013.6559650.

[38] R. Das and G. Tuna, "Packet tracing and analysis of network cameras with wireshark," *2017 5th International Symposium on Digital Forensic and Security (ISDFS)*, pp. 1–6, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:11222601.

[39] I. S. Thaseen, C. A. Kumar, and A. Ahmad, "Integrated intrusion detection model using chi-square feature selection and ensemble of classifiers," *Arabian Journal for Science and Engineering*, vol. 44, pp. 3357–3368, 2019.

[40] Y. Sugianela and T. Ahmad, "Pearson correlation attribute evaluation-based feature selection for intrusion detection system," in *2020 International Conference on Smart Technology and Applications (ICoSTA)*, IEEE, 2020, pp. 1–5.

[41] S. M. Kasongo and Y. Sun, "Performance analysis of intrusion detection systems using a feature selection method on the unsw-nb15 dataset," *Journal of Big Data*, vol. 7, pp. 1–20, 2020.

[42] Ortega Vázquez, Carlos and vanden Broucke, Seppe and De Weerdt, Jochen, "Hellinger distance decision trees for PU learning in imbalanced data sets," eng, *MACHINE LEARNING*, 2024, ISSN: 0885-6125. [Online]. Available: %7Bhttp://doi.org/10.1007/s10994-023-06323-y%7D.

[43] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, 2019.

[44] R. Thomas and D. Pavithran, "A survey of intrusion detection models based on nsl-kdd data set," *2018 Fifth HCT Information Technology Trends (ITT)*, pp. 286–291, 2018.

[45] V. Kumar, D. Sinha, A. K. Das, S. C. Pandey, and R. T. Goswami, "An integrated rule based intrusion detection system: Analysis on unsw-nb15 data set and the real time online dataset," *Cluster Computing*, vol. 23, pp. 1397–1418, 2020.

[46] M. Naser and A. Alavi, "Insights into performance fitness and error metrics for machine learning," *arXiv preprint arXiv:2006.00887*, 2020.