Hindawi
Security and Communication Networks
Volume 2023, Article ID 9845784, 1 page
https://doi.org/10.1155/2023/9845784

Retraction

Retracted: Lightweight Statistical Approach towards TCP SYN Flood DDoS Attack Detection and Mitigation in SDN Environment

Security and Communication Networks

Received 5 December 2023; Accepted 5 December 2023; Published 6 December 2023

This article has been retracted by Hindawi, as publisher, following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of systematic manipulation of the publication and peer-review process. We cannot, therefore, vouch for the reliability or integrity of this article.

Please note that this notice is intended solely to alert readers that the peer-review process of this article has been compromised.

Wiley and Hindawi regret that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] S. Batool, F. Zeeshan Khan, S. Qaiser Ali Shah et al., "Lightweight Statistical Approach towards TCP SYN Flood DDoS Attack Detection and Mitigation in SDN Environment," *Security and Communication Networks*, vol. 2022, Article ID 2593672, 14 pages, 2022.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 2593672, 14 pages
https://doi.org/10.1155/2022/2593672

Research Article

Lightweight Statistical Approach towards TCP SYN Flood DDoS Attack Detection and Mitigation in SDN Environment

Sehrish Batool,[1] Farrukh Zeeshan Khan,[1] Syed Qaiser Ali Shah,[1] Muneer Ahmed,[2] Roobaea Alroobaea,[3] Abdullah M. Baqasah,[4] Ihsan Ali,[5] and Muhammad Ahsan Raza[6]

[1]*Department of Computer Science, University of Engineering and Technology, Taxila, Pakistan*
[2]*School of Electrical Engineering and Computer Science (SEECS), National University of Sciences and Technology (NUST), Sector H-12, 44000, Islamabad, Pakistan*
[3]*Department of Computer Science, College of Computers and Information Technology, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia*
[4]*Department of Information Technology, College of Computers and Information Technology, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia*
[5]*Department of Computer System and Technology, Faculty of Computer Science and Information Technology, Universiti Malaya, Kuala Lumpur, Malaysia*
[6]*Department of Information Technology, Bahauddin Zakariya University, Multan 60000, Pakistan*

Correspondence should be addressed to Ihsan Ali; ihsanalichd@siswa.um.edu.my

Received 15 October 2021; Revised 29 December 2021; Accepted 4 January 2022; Published 25 January 2022

Academic Editor: Muhammad Arif

Distributed Denial of Service (DDoS) attack is known to be one of the most lethal attacks in traditional network architecture. In this attack, the attacker uses botnets to overwhelm network resources. Botnets can be randomly compromised computers or IoT devices that are used to generate excessive traffic towards the victim, and as a result, legitimate users cannot access the services. In this research, software-defined networking (SDN) has been suggested as a solution to fight DDoS attacks. SDN uses the idea of centralized control and segregation of the data plane from the control plane. SDN is more flexible, and policy implementation on the centralized controller is easy. SDN is now being widely used in modern network paradigms because it has enhanced security. In this work, an entropy-based statistical approach has been suggested to detect and mitigate TCP SYN flood DDoS attacks. The proposed algorithm uses a three-phased detection scheme to minimize the false-positive rate. Entropy, standard deviation, and weighted moving average have been used for intrusion detection. Multiple experiments were performed, and the results show that the suggested approach is more reliable and lightweight and has a minimal false-positive rate.

1. Introduction

The Internet has become a crucial part of homes and offices in the 21st century. Internet not only is being used for communication but also has made its place and value successful in businesses. Many business giants like Alibaba, Amazon, and Tencent Holdings rely on the Internet for their entire business. Internet is contributing to every sector, from schools [1] to health and homes. As per the Cisco Annual Internet Report [2], worldwide Internet users will reach 5.3 million by 2023, with over two-thirds of the global population having access. Because of its interdependence, the Internet is more vulnerable to intrusions and attackers. There is a history of Internet service disruptions. Users have been targeted over the information superhighway using a multitude of approaches. The first DDoS attack happened in 1996 when a DDoS attack rendered Panix, an Internet service provider, inaccessible for many days. As per the Cisco Annual Internet Report for 2018–2020, the number of DDoS attacks will be twofold by 2023. Table 1 is a list of the utmost famous DDoS attacks in recent years.

The system of cyberattack, in which many bots attack a single network, rendering it inaccessible for legitimate nodes, is

Table 1: Some of the famous DDoS attacks in history.

Victim	Year	Attack type	Attack rate
Google [3]	2020	Spoof	167 million pps
Amazon [4]	2020	Multivector DDoS	2.3 Tbps
GitHub [5]	2018	Memcached DDoS	126.9 million pps
NETSCOUT client	2018	Multivector DDoS	1.7 Tbps
Dyn	2016	Mirai malware	1.5 Tbps
Brian Kerbs	2016	Mirai malware	620 Gbps
BBC [6]	2015	Application based DDoS	600 Gbps
Hong Kong	2014	DDoS	500 Gbps
CloudFlare	2014	NTP protocol DDoS	400 Gbps
Spamhaus	2013	Spam DDoS	200 Gbps
Six Banks	2012	Various DDoS	63.3 Gbps

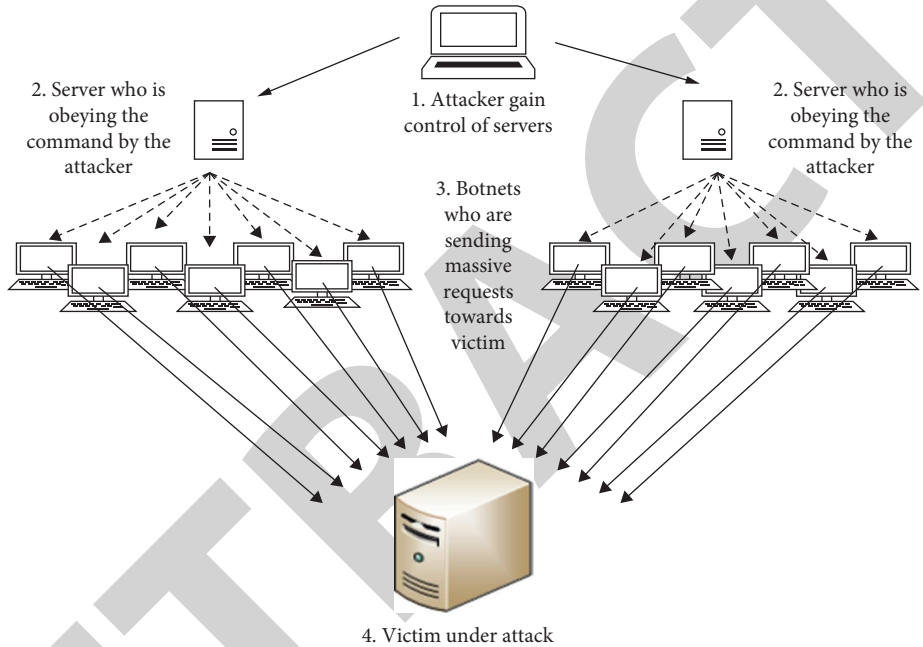

Figure 1: DDoS attack scenario.

a Distributed Denial of Service (DDoS) intrusion. DDoS attacks containing thousands of botnets [7] are commonly launched to interrupt web services. Although many defense strategies have been suggested to counter DDoS attacks [8], it is still a significant threat at the application level. The attacker's traffic is just like legitimate user traffic. These attacks can cause massive disruption and network congestion.

From Figure 1, the purpose of the DDoS attack is server exhaustion by generating excessive requests towards the server. The victim server will go offline, and as a result, its services will be unavailable for legitimate users. Botnets are also known as zombies and are random devices that are controlled remotely by an attacker using malware. Botnets are used to generate massive requests towards the server. The attacker usually has full control over the botnet, and many devices, including IoT devices, can be used as botnets. DDoS attacks can be of various types. DDoS attacks are generally classified based on attack technique and victim type. Few most common attack types have been classified into the following three categories in Figure 2.

The invader makes efforts to reside in all network's available bandwidth by pushing an enormous volume of traffic flow to the target of volume-based attacks, such as ICMP flood attackers. Botnets or other forms of amplification are used to send this traffic, causing network congestion [9].

In application-layer attacks, server resources like CPU and memory are exhausted by generating excessive ping requests towards the server. It is to detect DDoS attacks because the firewall cannot detect them, for example, HTTP GET flood attack [10].

Protocol-based attacks are designed to create a significant service interruption by using all available state table capacity. The goal of attacks like TCP SYN flood is to overwhelm the target and render it unusable for genuine users [11].

Transmission control protocol (TCP) is a secure application stratum protocol that establishes a secure connection before transmission. A three-way handshake is used to establish a secure connection between the sender and receiver.

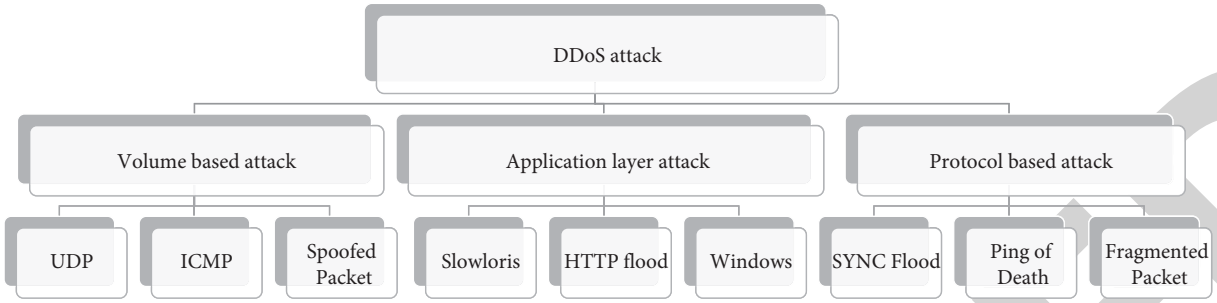

Figure 2: DDoS attack types.

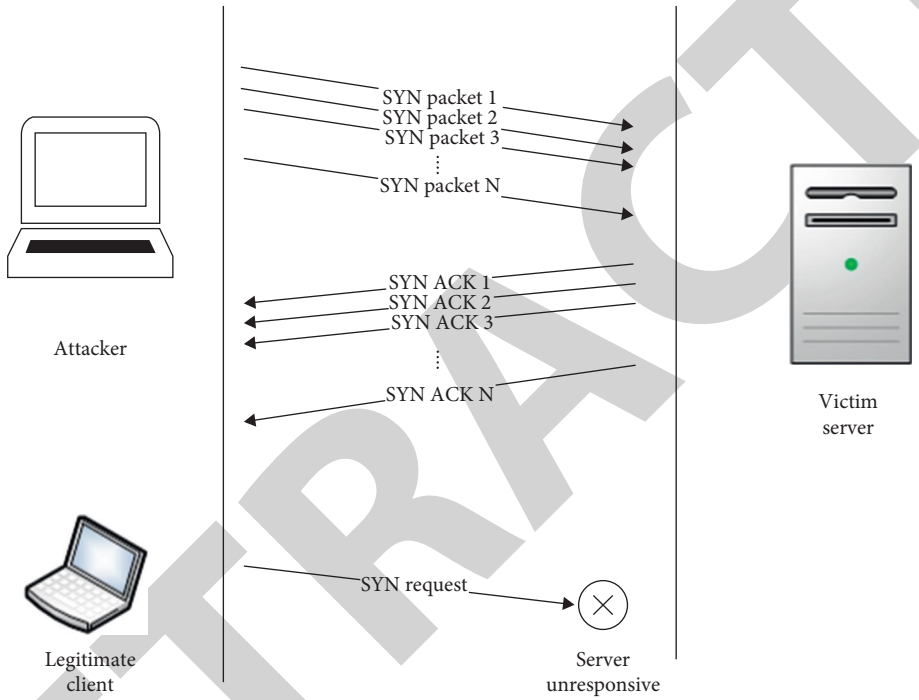

Figure 3: TCP SYN attack.

The receiver has a passive open connection. The sender has an active open connection. SYN packets are sent by the sender, requesting connection initiation, the receiver responds to the sender by distributing SYN-ACK packets to the SYN packet, and then the sender completes the connection by sending ACK packets again to the SYN-ACK packets. The invader takes benefit of the TCP three-mode handshake mechanism in TCP SYN flood attack and overloads the target with synchronization requests. TCP is a connection-oriented protocol, which creates a connection before transmitting data. Connection is established using a three-way handshake mechanism in which the client requests a connection from the server by sending synchronization packets. The server replies to the client by sending acknowledgments against synchronization packets. This mechanism of TCP is used to saturate server resources by sending excessive requests.

In Figure 3, the attacker generates a massive amount of SYN requests towards the server using botnets. The server thinks that the requests are genuine, so he replies with SYN-ACK packets and waits for the transmission to be started. However, the attacker leaves the connection half-open by never replying to SYN-ACK packets. These half-open connections make the server exhausted and bring it offline. If a legitimate user tries to access the server during this period, he gets no response as the attacker occupies the server resources. Many techniques for detecting and mitigating TCP SYN-based intrusions have been offered, and the approach provided in this study is based on SDN.

SDN stands for software-defined networking and is considered a next-generation network design. The primary characteristic of SDN is that it uses a centralized controller to decouple the record plane from the controller plane. A centralized controller makes it easy to implement security and rules inside the network. The controller works as a network supervisor who monitors the traffic coming to and going from the network. SDN was invented by ONF (open network foundation) in 2007. Definition of SDN provided by ONF says that "A network architecture in which the control plane controls the network flow and forwarding is separated

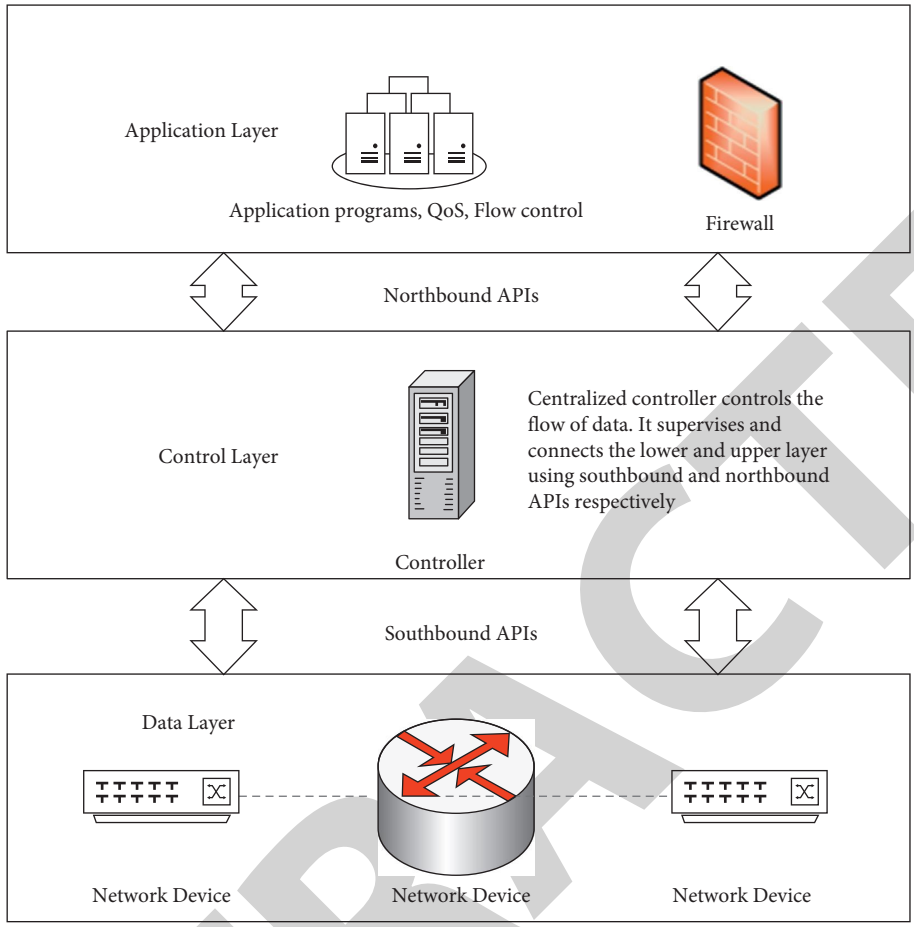

Figure 4: SDN architecture.

from control plane [12]." ONF gave an idea of SDN by segregating the control plane and data plane. OpenFlow protocol is used for SDN switch configuration and communication between SDN planes. OpenFlow switches are used in SDN. OpenFlow switches contain two key components, a secure communication channel and a flow table [13]; the secure channel allows controllers to deliver messages and packets to the switch, and the flow table instructs the switch how to process. The flow table has contained header, packets, process definition, and count of packets and bytes for each flow [14]. Packet forwarding, encapsulation, and dropping packets according to the security setting are done by OpenFlow. Data collection and firewall implementation are done by flow tables. Information of each layer is kept in the flow table so that the forwarding rules can be implemented easily. Further details of SDN architecture and working are given in Figure 4.

The architecture of SDN is explained in Figure 4. By separating the control plane forwarding and data plane, SDN ensures more flexibility, adaptability, security, and programmability. Communication apparatus, for instance, switches and routers, are said to be in the data plane in SDN, while the controller is in the control plane [15]. SDN has three planes that are separated by the southbound and northbound APIs. Three planes of SDN are as follows.

Application Plane. It includes all networking applications and software like load balancers, quality of service, firewall, and so on. Northbound APIs do communication between the controller and application plane.

Control Plane. The SDN's primary and controlling point is the controller. It acts as a network's brain, doing all required tasks to administer the network and networking devices. It also serves as a link between the network's hardware and software. The controller oversees putting policies and rules into action. It also keeps track of and filters all traffic entering and exiting the network.

Data Plane. Hardware infrastructure lies inside the data plane. The switches manage flow tables. SDN has OpenFlow switches; whenever a new packet enters inside the network, the flow rule implemented by the controller decides what to do with that packet.

For communication between planes, northbound and southbound application program interfaces (APIs) are used in SDN.

Northbound APIs. Policies are the rules that describe the network behavior. These rules are implemented by the controller. The controller tells application programs about hardware status, available resources, and network traffic. The

Table 2: Comparison of SDN with conventional network architecture [16].

Feature	Traditional network	SDN
Innovation	Limited hardware capacity and test environment.	Upgraded software provides fast deployment. Multitenants under centralized controller enhanced experimentations.
Configuration	Complex manual configuration.	A centralized controller provides an automated configuration.
Performance	Devices are statistically configured, having limited information.	Globally, dynamic configuration of devices has cross-layer information.
Management	It is hard to manage or change as every device needs changes separately.	Simplified management and improved delivery. Ensuring programmability by parting the control plane from the data plane.

controller communicates with the application plane using northbound APIs. Northbound APIs are also used by the application plane to find the best path, compute latency, and implement a security firewall.

Southbound APIs. Lower-level architectures like network switches and routers use southbound APIs to communicate with the controller. Flow policies and flow table entries are managed by the controller of the switches. The controller can change the switch behavior. How the traffic enters and how it will go outside the network will be decided by the controller. The comparison of SDN with traditional network architecture is given in Table 2.

These features of SDN make it popular among researchers and network providers. Response to changes is quick in SDN because of its cloud-like architecture. SDN has been chosen to implement DDoS attack detection scheme because of its architecture. This study aims to figure out how to identify Distributed Denial of Service attacks in SDN. For identifying TCP SYN flood attacks, a novel statistical technique called MMSA (Multimodular Statistical Approach towards DDoS Detection) has been proposed. The suggested approach is simple to use, has a low false-positive rate, and is lightweight. The proposed system is implemented via the POX controller in software-defined networks. For the discovery and mitigation of DDoS attacks, a lightweight code has been incorporated into the POX controller.

The paper organization has been done in the following manner: Section 2 explains the literature review, in Section 3, we have discussed the proposed technique, Section 4 describes the experimental scheme and results, and the conclusion has been discussed in Sections 5 and 6.

2. Literature Review

Multiple solutions have been suggested to spot and mitigate DDoS attacks. The authors in [17] have suggested a discovery and mitigation of the DDoS attack approach through machine learning. For detection, they have used a support vector machine (SVM). In SVM, six-tuple characteristics' information is received from the switch, which is classified by an algorithm to detect attack traffic. This technique provides high precision and a low false alarm rate. However, the computational time of this method is high.

In communication networks, entropy-based algorithms [18] can also be employed to spot DDoS attacks. This statistical method can spot various attacks by detecting the unpredictability of arriving traffic, including DDoS [15]. The higher value of entropy indicates more randomness of traffic flow, and a small value of entropy means more reliable traffic flow. The entropy-based model was used to detect TCP SYN flood DoS attacks in [19]. The model was tested using a variable window size of data generated by the OpenFlow switch and controller. In [20], the entropy-based detection method is used along with PSO-BP neural network in the SDN environment. The predetection is accomplished on switches by dividing them into normal and abnormal using entropy. The results state that this method provides better performance, but this can only be used to detect data layer attacks and cannot be used to control layer DDoS attack detection. In another study [21], the concept of cross entropy was introduced. Cross entropy depends on the packet arrival rate in the SDN. The limitation of this work is single factor validation. If we use the packet arrival rate for attack detection on a busy server, it can result in a higher false-positive rate.

In research [22], entropy has been used for control layer slow DDoS attack detection. Their main contribution is slow DDoS attack detection and comparison of their results with Shannon's results. In [23], the entropy-based detection method is used for UDP flood attack detection in the SDN surroundings. The limitation of their work was small (i) window size, that can cause computational overhead, and a (ii) single factor detection scheme can generate the higher value of false-positive rate. References [24, 25] have used the entropy-based method to notice the attack. Entropy-based detection schemes depend on network feature distribution such as source IP, destination IP, source port, or/and destination port to detect the malicious behavior of network traffic. A predefined threshold value of entropy is used to detect abnormal traffic flow [26]. The critical factor of this method is a threshold value; this whole detection method solely depends on a threshold value. The slight difference in the threshold values can lead to an extraordinary rate of false positives. Due to this, most modern research uses other algorithms along with entropy.

The security benefits of SDN architecture cannot be overstated. For example, data derived from network traffic examination and anomaly recognition can be continuously sent to the central controller. The central controller can use the broad network infrastructure view to correlate and evaluate this type of network feedback [27, 28]. New security measures are created due to this technique to prevent attacks from spreading across the network. SDN's greater programmability and performance, together with network visibility, can improve network security management and

containment. Despite all benefits that SDN technology offers, it can be exposed to many types of attacks, for instance, anomaly attacks, intrusion attacks, Denial of Service (DoS), and assaults. Furthermore, while the architecture supports network security, the SDN programmability feature could reduce energy consumption by the network infrastructure and devices.

In paper [29], they have used bioinspired anomaly-based attack detection using a Bat algorithm. First, they extracted features for the identification of attacks. The next phase is the customization of the Bat algorithm for training and testing. They also compared their detection technique with ARTP and FCAAIS techniques and showed that the Bat algorithm increased accuracy with minimal process complexity. In previous research, the solution was projected for TCP. There are two types of SYN flood attacks: host-based solutions and network-based solutions. End-host TCP implementation, such as server-side rate limiting [30, 31], raising TCP backlog, or altering the methods and data structures required for connection creation, such as TCP SYN Cookies, is the focus of host-based solutions [32], TCP SYN cache, limiting the number of SYN-RECEIVED timers [33]. The common feature of these solutions is that they do not detect malicious traffic and legitimate traffic from the flow but improve the tolerance capacity of the controller during the attack. In [34], researchers proposed a multilevel framework for DDoS mitigation, which included fog computing, cloud computing, and edge computing levels. They used SDN for implementation. Simulation results showed that the combination of three levels of computing would help us solve the DDoS attack issues in IoT devices.

In December 2018, Thomas Lukaseder et al. [30] presented a detection and mitigation scheme for slow HTTP attacks. They presented several concepts based on lightweight flow-based investigation of network traffic to help identify and exclude attackers from the network. The system's limitation was its accuracy, which was not high enough, and it was only able to detect HTTP-based attacks. In [31], the authors generated an ICMP flood attack on SDN based network, then monitored the traffic, and applied mitigation on the controller. They used Sflow for monitoring traffic generated on the mininet emulator. Sflow analyzer may be embedded into the controller, and its communication overhead requires just 0.02 percent of a 10 GB Ethernet connection, response discovery time 5 secs, and control time equals 10 secs, according to their research. It is the best solution for preventing DDoS attacks, but adding additional switches would need more effort to manage the complicated network.

Authors in [5] introduced Cochain-SC, a Blockchain-based approach with two layers of mitigation: interdomain and intradomain combination. In the context of SDN, there are three mitigation schemes: (1) intraentropy to assess data unpredictability, (2) intrabay-based approach to classify attack traffic, and (3) intradomain mitigation that is a strategy for preventing intradomain attacks. In this work, they have defined the flow as seven tuples.

Zhijun et al. proposed another DDoS attack detection algorithm in [35]. This paper investigates the slow TCAM Exhaustion attack, an LR_DDoS attack, and presents a multifeatured detection scheme based on Factorization Machine (FM) machine learning method. They also suggested a protection scheme based on dynamic flow rule detection. Certain features must first be retrieved to identify an LR-DDoS attack. Compared to controller-based detection approaches, which sample the entire flow table, extracting these features allows for individual sampling of flow rules. These features are used as a training dataset for FM mode, and the model's parameters are optimized using the Stochastic Gradient Descent (SGD) algorithm. The trained FM algorithm model then tests the obtained flow rule. The flow rule is added to the Attack Database and Delete Database if it classifies it as malignant. At the first deletion interval, the flow rule is dynamically erased. The number of flow rules at the switch under attack will be reduced because of this.

In [36], researchers have used the bioinspired anomaly-based HTTP Flood attack detection technique. They have used a cuckoo search technique named after a bird because of its search method for laying eggs. Their research has three stages. The initial stage is defining feature metrics that are done on a fixed time interval. In the next stage, these features are optimized as usual or flooded using correspondence assessment by the Jaccard index. The third stage is the customization of cuckoo search in hierarchal order to train and examine. This customization promises a selection of optimal attributes and provides detection accuracy as well as minimal computational overhead.

AVANT-GUARD is a system to spot and mitigate DDoS attacks proposed by Seungwon Shin et al. in [37]. This method was suggested by the fact that the attacker is never attempting to complete a TCP handshake. The controller never receives TCP-ACK packets. AVANT-GUARD is used to protect switches. It serves as a stand-in server that takes TCP connections. The module is connected to the actual server if the TCP-ACK packet is received, and a connection has been established. This algorithm can have a high false-positive rate for the slow legitimate client. In [38], this paper has used bioinspired anomaly-based attack detection using a Bat algorithm. First, they extracted features for the identification of attacks. The next phase is the customization of the Bat algorithm for training and testing. They also compared their detection technique with ARTP and FCAAIS techniques and showed that the Bat algorithm increased accuracy with minimal process complexity.

Another technique [39] is using multiple controllers to detect and avoid DDoS attacks. As the name suggests, this technique is about load balancing and protects the network itself. However, using multiple controllers will add extra cost and network complexity. Several techniques have been proposed, and each method has its pros and cons. For example, machine learning-based techniques provide high detection accuracy, but it requires high computational time. Comparative analysis for various techniques is discussed in Table 3.

From Table 3, we can generalize the pros and cons of different attack detection schemes. ML-based techniques provide a low false-positive rate with high detection

Table 3: Comparative study of various DDoS detection strategies.

Ref. no.	Approach	Method	Benefits	Limitations
[40]	Anomaly-based detection	Entropy, source IP index, and packet rate	Low computational time, low false-positive/negative rate, and high detection throughput	Low accuracy and low adaptability
[41]	Machine learning	DNN	High accuracy	High computational time
[42]	Statistical	Used FGPA	Nine types of DDoS attacks can be detected based on incoming traffic	High detection time, computational complexity, and low flexibility
[43]	Rate limiting	FlowSec	Low computational time and high detection throughput	Low accuracy and high false-positive rate
[44]	Statistical	Switch statistics	Low computational time, High accuracy, and flexibility	High false- positive/negative rate and complexity
[45]	Machine learning	LSTM + CNN	High detection accuracy and minimal false-positive rate	Memory consumption and computational complexity are high
[46]	Hybrid	Long short-term memory + fuzzy logic	High accuracy, flexibility, and minimum damage	Computational complexity

accuracy, but it requires high computation time and complexity. Statistical approach towards DDoS detection provides no computational overhead [47], faster attack detection in large networks, and low computational time. In short, the statistical approach is fast, easy to implement, and flexible. It has no extra implementation cost. Therefore, they are known to be lightweight approaches towards DDoS attack detection.

In this research, a statistical-based approach along with entropy named MMSA (Multimodular Statistical Approach towards DDoS Detection) has been proposed. The MMSA method is proposed and tested in this research, which is a statistical approach towards DDoS attack detection based on multiple threshold values calculated in real time, which minimizes the controller overhead and false-positive rate.

3. Multimodular Statistical Approach towards DDoS Detection

The proposed method Multimodular Statistical Approach towards DDoS Detection (MMSA) consists of multiple phases that work together to detect and mitigate specifically TCP SYN flood attacks. Using multiple phases for attack detection helps in minimizing the false positive. As in larger networks, it is difficult to differentiate between normal traffic surge or attack flow coming towards the traffic. Using three parameters for testing will reduce the false positive.

Weighted moving average and standard deviation have been used for detection along with entropy. Standard deviation and moving average have been used to determine the level one threshold. If the packet_ count is greater than ∂_n, an alarm will be triggered, and the algorithm will move towards the second phase of detection. For entropy calculation, we have used the destination IP of the sending device, and it will be calculated in a fixed window size of 50. The threshold value for entropy is 1. The algorithm for the proposed method is given in Algorithm 1. Basic TCP parameters like packet counter per second, source, and destination IP to measure the entropy and threshold values have been used to ensure speed and minimum computation time. Table 4 explains the details of the algorithm used by notation.

Table 4: Features and parameter's notation for algorithm.

Sr. no.	Feature and parameter	Notation
1	Source IP and destination IP	IP_{src}, IP_{dst}
2	Exponentially weighted moving average	$\overline{X_n}$
3	Entropy	E(H)
4	Threshold 1	∂_n
5	Threshold 2	γ_n
6	Standard deviation	S_n
7	Packet ratio	P_r
8	Temporary address database	TAD
9	Permanent address database	PAD
10	Flow stat database	FSD

Algorithm 1: Multimodular Statistical Approach towards DDoS Detection.

3.1. TCP SYN Flood Detection. Statistical-based three-phased attack detection model MMSA has been used to minimize the false-positive rate. In phase 1, the time series-based detection method was used. In this method, the algorithm will monitor the degree of incoming SYN packets in a fixed window size Δw. Incoming packets will be monitored and stored in a temporary table named Temporary Address Table (TAT.), and start_time will be set to 0. These packets will be compared to a threshold value ∂_n at the conclusion of window Δw, and if the rate of SYN packet is higher than ∂_n, the algorithm will trigger an attack warning, and detection phase II will be started.

$$\begin{aligned}\text{EWMA} &= \overline{X_n} \\ &= \overline{X_{n-1}} + \alpha\left(X_n - \overline{X_{n-1}}\right),\end{aligned} \tag{1}$$

$$\begin{aligned}\text{Standard Division} = \text{Sd} &= S_n \\ &= \sqrt{\alpha * \left(X_n - \overline{X_{n-1}}\right)^2 + (1-\alpha)\left(S_{n-1}\right)^2},\end{aligned} \tag{2}$$

$$\partial_n = \overline{X_n} + kS_n. \tag{3}$$

(1) Start
(2) **DATA**: TCP_packets, IP_{src}, IP_{dst}, $\overline{X_n}$, S_n, δ_n,E(H), P_r, EWMA (P_r), $S\ D\ (P_r)$, γ_n , current_time, end_time, PAD, TAD, Packet_count, FSD.
(3) **Output:** Allow the flow coming from a legitimate host block if attacked.
(4) **if** ⟵ TCP_Packet
(5) Add in TAD
(6) **if** ⟵ current_time > End_time
(7) Calculate X_n, S_n
(8) Calculate δ_n, δ_n, $E(H)$, P_r, EWMA (P_r), $S\ D\ (P_r)$, γ_n
(9) Update end_time = current_time + 3
(10) Update PAD
(11) Packet_count = 0
(12) **else**
(13) **if** ⟵ Packet_count > ∂_n
(14) **If** IP in FSD is in PAD
(15) end_time = current_time + 3
(16) **else**
(17) **If** ⟵ $P_r > \gamma_n$ && $E(H) < 1$
(18) **Attack detected**
(19) **Mitigate**
(20) **else**
(21) end_time = current_time + 3
(22) **end If**
(23) **end If**
(24) **else**
(25) Packet_ count + = 1
(26) **end If**
(27) **END**

Algorithm 1: TCP SYN flood attack detection and mitigation in SDN environment.

Using equations (1) and (2), the value of the weighted moving average and standard deviation for a window size will be calculated. Here, it is necessary to know that α is a tuning parameter. In this algorithm, the value of α is set to 0.05, which provides equal weight to recent and older data. K used in the equation is a constant. Equation (3) is used to calculate the value of threshold 1. ∂_n is the sum of EWMA and standard deviation and value of ∂_n will be compared to the packet count P_n. If the value of ∂_n is less than P_n, the flow is suspicious and detection phase II will start detection. In equation (4), the value of threshold 1 (∂_n) will be compared to the packet count (P_n).

$$\partial_n < P_n. \tag{4}$$

In detection phase II, the entropy of incoming packets will be measured using equation (5) and compared to a predefined threshold value, which is set to 2. Entropy is the degree of randomness in a given data, so less value of entropy means a low rate of randomness, whereas a high value means a higher level of randomness. We assume that if the value of entropy is less than 2. The traffic can be malicious, and phase 3 will get started for detection verification.

$$E(H) = \sum_{i=1}^{n} \text{pilogpi}. \tag{5}$$

In step 3, more investigation is carried out by connecting the source IPs of all flows to a database of legitimate source IPs. Equation (6) will be used to calculate the ratio of the single packet flow to the total number of flows, which will be compared to another threshold, say $\_n$. When the ratio of a single packet flow exceeds $\_n$, an attack is detected.

$$P_r = \frac{(C_{\text{single}})}{(C_{\text{total}})} * 100, \tag{6}$$

$$\begin{aligned} \text{EWMA}\ (P_r) &= \overline{P_{rn}} \\ &= \overline{P_{rn-1}} + \alpha\left(P_{rn} - \overline{P_{rn-1}}\right), \end{aligned} \tag{7}$$

$$S\ D\ (P_r) = \sqrt{\alpha * \left(P_{rn} - \overline{P_{rn-1}}\right)^2 + (1-\alpha)\left(SD_{n-1}\right)^2}. \tag{8}$$

Equations (7) and (8) are used to calculate the exponential moving average and standard deviation for a single packet ratio.

$$\gamma_n = \text{EWMA}\ (P_r) + S\ D\ (P_r). \tag{9}$$

Now,

$$\gamma_n < P_r. \tag{10}$$

Equation (10) compares the single packet ratio P_r, to the threshold γ_n, and if the threshold value is less than the single packet ratio, it will confirm that the incoming flow is an attack flow, so the mitigation phase will start. Here, it is to be noted that the algorithm will allow the traffic flow to minimize the false-positive rate; this period is set to 3

Table 5: Parameters and values for experimental setup.

Sr. no.	Parameters	Values
1	No of switches	2
2	No. of hosts	53
3	No. of target	1
4	No. of legitimate traffic nodes	50
5	No. of attacking hosts	3
6	Detection time	3 seconds
7	Monitoring duration	300 seconds
8	Generated attack traffic	TCP sync
9	Window size	64
11	Traffic generator	Hping3
11	Monitoring tool	Netdata Agent
12	Entropy threshold	1
13	Entropy window	50

Table 6: Parameters used for evaluation.

S. no.	Parameter	Description
1	Bandwidth	Bandwidth consumed by ingress flow.
2	TCP packets	The number of TCP packets received.
3	Entropy	Entropy is the degree of random numbers, and low entropy means a higher degree of randomness and vice versa.
4	Softirq	High-priority software processes are handled by software interrupts known as softirqs.
5	Softnet	Softnet statistics state the number of packets processed.

seconds. After this time, the algorithm will again mitigate all malicious packets.

3.2. Attack Mitigation. For SYN attack mitigation, the sinkhole strategy has been implemented. Port five is set as a sinkhole, and the attack flow will be given the lowest priority and migrated towards port five, whereas legitimate traffic flow will have the highest priority.

4. Experimental Setup

MMSA has been implemented on a VMware with 8 GB RAM, Ubuntu 16.04 LTS OS. Mininet [48] emulator has been used for SDN simulation, and POX controller has coupled to a network containing two switches. Forty-three hosts are connected to the switches. Host h52 acts as a server, whereas four hosts, h1, h2, h3, and h4, are acting as clients; three hosts, h13, h23, and h33, are acting as attacking devices. Hping3 [49] tool has been used for attack traffic, and Netdata Agent [50] is used for traffic flow monitoring and analysis. Netdata is a real-time traffic monitoring tool that is installed on a computer with 4 GB RAM and core i3. Details of the parameters and values of each parameter have been explained in Table 5.

Figure 5 explains the topology used in the experimental setup. The victim server is connected to switch 2 and is being attacked by the attacker via switch 1. Switch 1 is linked to switch 2. The controller is connected to the network using switch 1.

For analysis, two types of traffic pattern scenarios are used. One is traffic of legitimate flow, and the other is attack traffic. Attack traffic is further divided into two scenarios: one is attack traffic without mitigation algorithm, and the other is attack traffic with mitigation.

4.1. Legitimate Traffic. The MMSA DDoS detection scheme was tested by generating legitimate traffic flows. SDN having 16 nodes and three connections with legitimate traffic flow were used.

4.2. Attack Traffic. To generate an attack, we have used the hping3 tool. The proposed methodology was tested by using three nodes as attacking hosts. Table 6 explains the parameters used for the observation of controller behavior under attack as well as under normal flow.

5. Results and Discussion

In the first experiment, the normal flow of TCP packets was monitored for 300 seconds. Our second experiment was monitoring traffic flow for attack traffic without mitigation. Purposed mitigation technique was tested in the third experiment. TCP traffic in the legitimate flow was coming from three different host nodes, h7, h8, h9, and h10. Three nodes, h13, h23, and h33, were working as attacking hosts. Following are the results for various scenarios.

In Figure 6, the suffix N is used for legitimate traffic under normal flow, A is for attack flow, and M is for attack mitigation algorithm. In Figure 6, for legitimate flow, we can see that softirqs are negligible. Moreover, transmitted (NET_TX_N) and received (NET_RX_N) softirqs are the same. For attack traffic, the following command has been used on four different host machines.

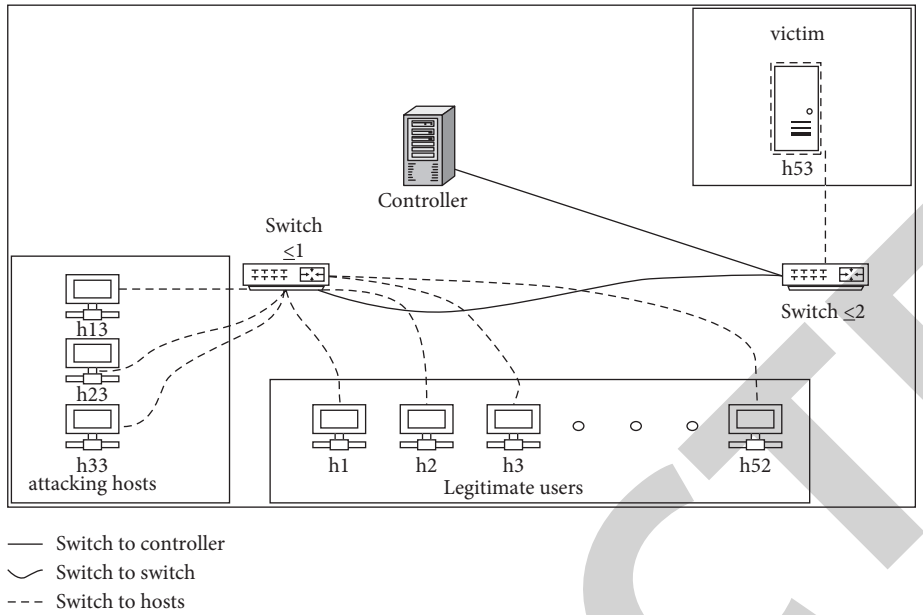

Figure 5: Experimental setup.

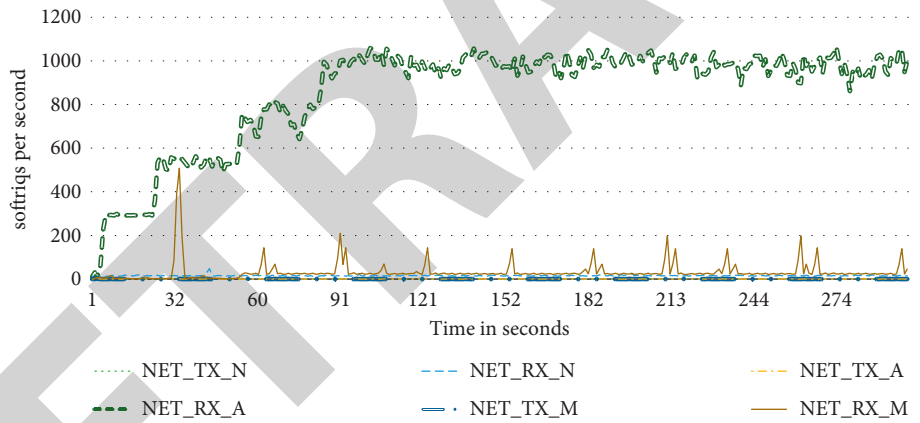

Figure 6: Softirqs per second.

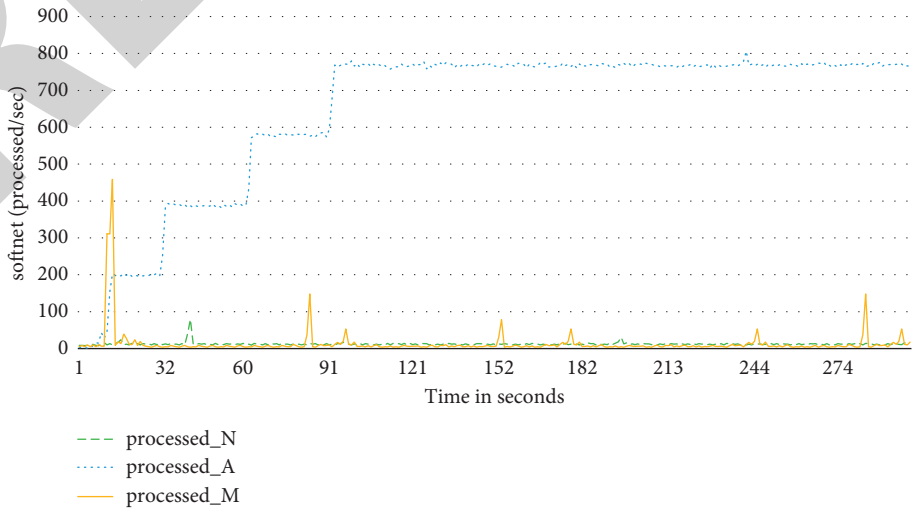

Figure 7: Softnets per second.

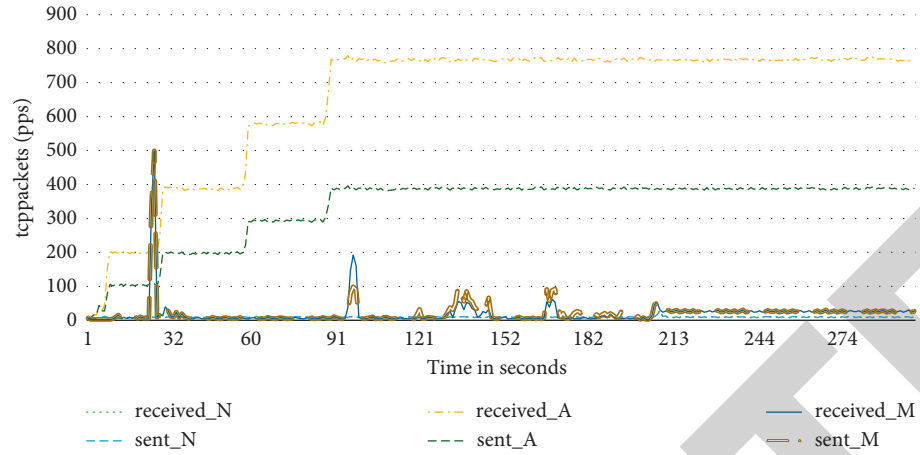

Figure 8: TCP packets per second.

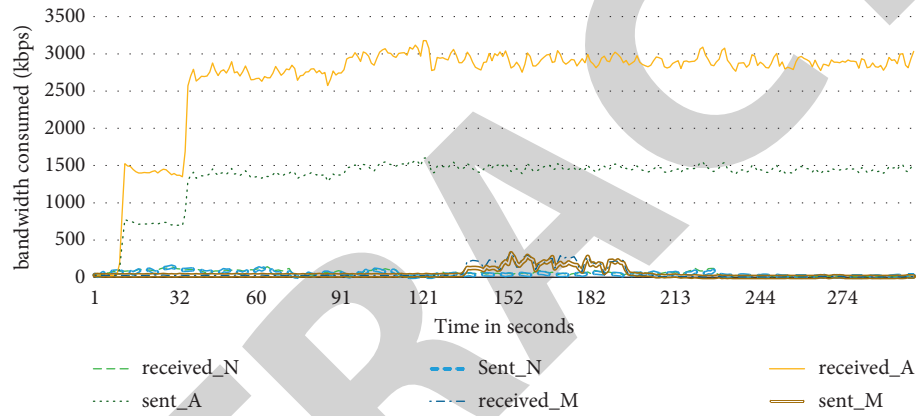

Figure 9: Bandwidth consumed per kbps.

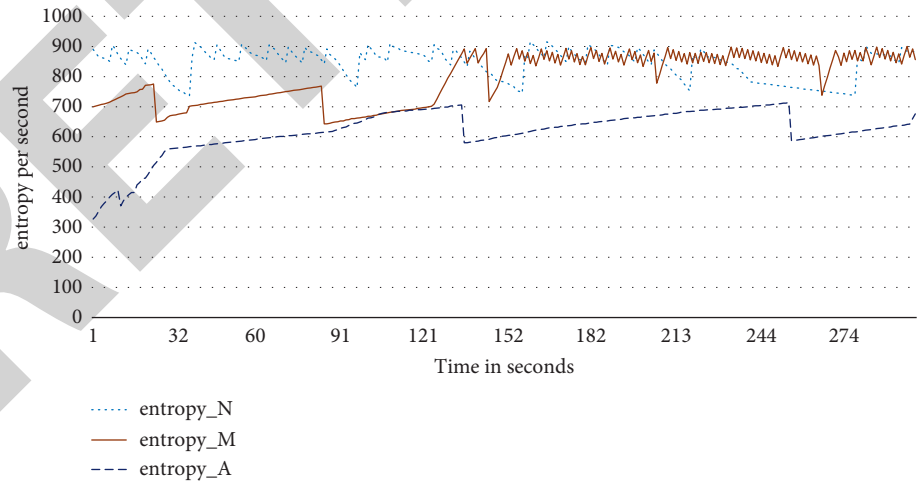

Figure 10: Entropy per second.

Sudo hping3 -P 80 -I u10000 -A SOURCE IP -S DESTINATION IP

After 2 seconds, the first attacking device was attacked. There, we saw a rise in softirqs received (NET_RX_A). The second device attacked 10 seconds after the first device, and the third device started attacking after 20 seconds. Here, it can be seen that the transmission softirqs (NET_TX_A) level is nearly zero. To check the performance of our algorithm, the attack started after 32 seconds, and we saw a slight bump in softirqs received (NET_RX_M). Attack was detected and mitigated after 3 seconds at 35. Our algorithm checks for legitimate traffic every 30 seconds for 3 seconds, so we can

see a slight increase in softirqs received (NET_RX_ M), while the amount of softirqs transmitted (NET_TX_M) remains constant throughout the experiment.

Softnets processed per second have been analyzed for 300 seconds for legitimate flow, attack flow without mitigation, and attack flow with mitigation. We can see from Figure 7 that softnets began to rise by 200 per attacking host for attack traffic (processed_A), which was almost zero for normal/legitimate flow (processed_N). Whereas initially, the value rose for attack flow with the mitigation algorithm, it was detected and mitigated in 3 seconds, and the value of softnet returned to normal. Here, it is essential to mention that a few packets of attack flow were still in the pipe, so it took more time for the botnets to return to normal. Our algorithm allows traffic for 3 seconds after every 30 seconds to avoid false positive so a slight increase in the value of softnets can be viewed from time to time.

Several TCP packets coming towards and going out of the device were also monitored and analyzed using the Netdata Agent. From Figure 8, we can see that, under standard flow, packets were coming at 14 packets to 24 packets per second and the amount of incoming (received_N) and outgoing (sent_N) was equal during legitimate traffic flow.

The results of bandwidth consumption are shown in kbps in Figure 9. The figure demonstrates that, for the standard TCP flow, the overall bandwidth consumed was about 100 kilobits per second. Values of received_N and sent_N remained the same during the whole experiment. As quickly as the attack started, the bandwidth consumption sharply increased, reaching almost a rate of 3 Mbps, consuming the whole network bandwidth, whereas for the attack with mitigation, we saw an increase up to 150 kilobits per second, but the bandwidth consumption became normal as soon as mitigation started. Then, a slow increase can be seen at t of 135 seconds; the mitigation algorithm allows traffic for a few seconds to avoid false positives.

The degree of randomness in traffic flow is measured using entropy. A higher value of entropy means there is no attack, and traffic on the network is legitimate. On the contrary, smaller values of entropy prove that the incoming traffic is malignant. From Figure 10, we can see that the value of entropy (entropy_N) is about 900 per second for the expected traffic. The value of entropy was decreased up to 300 per second for attack traffic without mitigation (entropy_A). After 20 seconds, we can see that the value raised to 570 packets per second, then the value further increased and reached up to 600 at $t = 91$ seconds, and it kept increasing until $t = 132$ seconds. Here, it reached 700 per second. After that, the value remained to fluctuate between 600 and 700 per second. For the attack flow with mitigation, we can analyze that the value of entropy was 700 per second, which increased up to 790 per second at $t = 12$ seconds.

6. Conclusion

This paper has proposed a lightweight statistical approach towards TCP SYN flood DDoS attack detection and mitigation in the SDN environment. The objective of this technique was to design and propose a lightweight and practical method for the diagnosis and diminution of TCP SYN flood attacks in the SDN. This technique has three modular approaches to notice and mitigate DDoS attacks. The method used multiple phases to accurately detect the attack. The limitation of the statistical approach is its dependency on selected parameters.

The MMSA method was tested and analyzed several times. Results illustrate that the algorithm can accomplish efficiently in different attack scenarios against TCP SYN flood attacks. The suggested method provides lightweight code for DDoS attack detection and mitigation scheme in SDN architecture. Three phases ensure the minimal false-positive rate and high accuracy, and it does not block the whole network traffic. It is aimed to try to extend our method to other attacks for future work, for example, ICMP and spoofing attacks.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors are grateful to the Taif University Researchers Supporting Project (no. TURSP-2020/36), Taif University, Taif, Saudi Arabia. This research work was partially supported by the Faculty of Computer Science and Information Technology, University of Malaya, under the Postgraduate Research Grant (PG035-2016A).

References

[1] U. Salih, "Educational uses of Internet in the world and Turkey (a comparative review)," *The Turkish Online Journal of Distance Education*, vol. 4, no. 3, 2003.

[2] Z. Abou El Houda, A. S. Hafid, L. Khoukhi, and S. C. Cochain, "Cochain-SC: an intra- and inter-domain ddos mitigation scheme based on Blockchain using SDN and smart contract," *IEEE Access*, vol. 7, pp. 98893–98907, 2019.

[3] P. Muncaster, "Google reveals it was hit by 2.5tbps DDoS," 2020, https://www.infosecurity-magazine.com/news/google-reveals-it-was-hit-by/.

[4] M. Pinho, "Aws Security Blog," 2021, https://aws.amazon.com/blogs/security/aws-shield-threat-landscape-review-2020-year-in-review/.

[5] S. Kottler, "The Github Blog," 2018, https://github.blog/2018-03-01-ddos-incident-report/.

[6] T. Mahjabin, Y. Xiao, W. Jioang, and G. Sun, "A survey of distributed denial-of-service attack, prevention, and mitigation techniques," *International Journal of Distributed Sensor Networks*, vol. 13, no. 12, Article ID 1550147717741463, 2017.

[7] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013.

[8] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM - Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[9] N. N. Tuan, P. H. Hung, N. D. Nghia, N. V. Tho, T. V. Phan, and N. H. Thanh, "A DDoS attack mitigation scheme in ISP networks using machine learning based on SDN," *Electronics*, vol. 9, no. 3, p. 413, 2020.

[10] C. Tang, A. Tang, E. Lee, and L. Tao, "Mitigating HTTP flooding attacks with meta-data analysis," in *Proceedings of the 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*, August 2015.

[11] N. Ravi, S. M. Shalinie, C. Lal, and M. Conti, "AEGIS: Detection and Mitigation of TCP SYN Flood on SDN Controller," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 745–759, 2020.

[12] O. Tr, "Principles and Practices for Securing Software-Defined Networks," *Open Networking Foundation*, Palo Alto, CA, USA, 2015.

[13] A. M. Alshnta, M. F. Abdollah, and A. Al-Haiqi, "SDN in the home: a survey of home network solutions using Software Defined Networking," *Cogent Engineering*, vol. 5, no. 1, Article ID 1469949, 2018.

[14] Y. Tang, Q. Chen, J. Xu, and Y. Chen, "Review on the 10-year development of software defined networks," *The Frontiers of Society, Science and Technology*, vol. 2, no. 11, 2020.

[15] M. Abdullah, N. Al-awad, and F. Hussein, "Implementation of entropy-based distributed denial of service attack detection method in multiple POX controllers," *Review of Computer Engineering Studies*, vol. 6, no. 2, pp. 29–38, 2019.

[16] T. Ubale and A. K. Jain, "Survey on DDoS attack techniques and solutions in software-defined network," in *Handbook of Computer Networks and Cyber Security*Springer, New York, NY, USA, 2020.

[17] J. Ye, X. Cheng, J. Zhu, L. Feng, and L. Song, "A DDoS Attack Detection Method Based on SVM in Software Defined Network," *Security and Communication Networks*, vol. 2018, Article ID 9804061, 8 pages, 2018.

[18] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[19] R. N. Carvalho, J. L. Bordim, and E. A. P. Alchieri, "Entropy-based DoS attack identification in SDN," in *Proceedings of the 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2019.

[20] J. David and C. Thomas, "DDoS attack detection using fast entropy approach on flow- based network traffic," *Procedia Computer Science*, vol. 50, no. 4, pp. 30–36, 2015.

[21] Z. Zhu, L. Wang, L. Xia, and S. Cao, "Flow performance analysis of software defined networking based on cross entropy," in *Proceedings of the 2015 8th International Congress on Image and Signal Processing (CISP)*, October 2015.

[22] K. S. Sahoo, B. Sahoo, M. Vankayala, and R. Dash, "Detection of control layer ddos attack using entropy metrics in sdn: an empirical investigation," in *Proceedings of the 2017 Ninth International Conference on Advanced Computing (ICoAC)*, December 2017.

[23] Z. Liu, Y. He, W. Wang, and B. Zhang, "DDoS attack detection scheme based on entropy and PSO-BP neural network in SDN," *China Communications*, vol. 16, no. 7, pp. 144–155, 2019.

[24] R. Swami, M. Dave, and V. Ranga, "Defending DDoS against software defined networks using entropy," in *Proceedings of the 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*, April 2019.

[25] P. Kumar, M. Tripathi, A. Nehra, M. Conti, and C. Lal, "SAFETY: early detection and mitigation of TCP SYN flood utilizing entropy in SDN," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1545–1559, 2018.

[26] N. Z. Bawany, J. A. Shamsi, and K. Salah, "DDoS attack detection and mitigation using SDN: methods, practices, and solutions," *Arabian Journal for Science and Engineering*, vol. 42, no. 2, pp. 425–441, 2017.

[27] A. Aleroud and I. Alsmadi, "Identifying DoS Attacks on Software Defined Networks: A Relation Context Approach," in *Proceedings of the NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*, April 2016.

[28] M. Ilyas, "A survey of DDoS attack detection strategies in cloud," *VFAST Transactions on Software Engineering*, vol. 8, no. 1, pp. 55–63, 2020.

[29] I. Sreeram and V. P. K. Vuppala, "HTTP flood attack detection in application layer using machine learning metrics and bio inspired bat algorithm," *Applied computing and informatics*, vol. 15, no. 1, pp. 59–66, 2019.

[30] T. Lukaseder, B. Erb, L. Maile, and F. Kargl, "Sdn-assisted network-based mitigation of slow ddos attacks," in *Proceedings of the International Conference on Security and Privacy in Communication Systems*, August 2018.

[31] B. H. Lawal and A. Nuray, "Real-time detection and mitigation of distributed denial of service (DDoS) attacks in software defined networking (SDN)," in *Proceedings of the 2018 26th Signal Processing and Communications Applications Conference (SIU)*, May 2018.

[32] R. Mohammadifar, E. Lotfi, A. Khosravi, and S. Nahavandi, "A heterogeneous defense method using fuzzy decision making," in *Proceedings of the 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, July 2017.

[33] S. Kumar, S. Member, and R. S. Reddy Gade, "Evaluation of microsoft windows servers 2008 & 2003 against cyber attacks," *Journal of Information Security*, vol. 06, no. 2, pp. 155–160, 2015.

[34] Q. Yan, W. Huang, X. Luo, Q. Gong, and F. R. Yu, "A multilevel DDoS mitigation framework for the industrial Internet of Things," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 30–36, 2018.

[35] W. Zhijun, X. Qing, W. Jingjie, Y. Meng, and L. Liang, "Low-rate DDoS attack detection based on factorization machine in software defined network," *IEEE Access*, vol. 8, pp. 17404–17418, 2020.

[36] K. Munivara Prasad, A. Rama Mohan Reddy, and K. Venugopal Rao, "BIFAD: bio-inspired anomaly based HTTP-flood attack detection," *Wireless Personal Communications*, vol. 97, no. 1, pp. 281–308, 2017.

[37] S. Shin, V. Yegneswaran, G. GU, and P. A. Porras, "Avant-guard: scalable and vigilant switch flow management in software-defined networks," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, Berlin, Germany, November 2013.

[38] K. M. Prasad, A. R. M. Reddy, and K. V. Rao, "BARTD: bio-inspired anomaly based real time detection of under rated App-DDoS attack on web," *Journal of King Saud University - Computer and Information Sciences*, vol. 32, no. 1, pp. 73–87, 2020.

[39] N. Dayal and S. Srivastava, "Leveraging SDN for early detection and mitigation of DDoS attacks," in *Proceedings of the*

International Conference on Communication Systems and Networks, January 2018.

[40] N. Hoque, H. Kashyap, and D. K. Bhattacharyya, "Real-time DDoS attack detection using FPGA," *Computer Communications*, vol. 110, pp. 48–58, 2017.

[41] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for Internet of Things," *Future Generation Computer Systems*, vol. 82, pp. 761–768, 2018.

[42] B. Nagy, P. Orosz, T. Tóthfalusi, L. Kovács, and P. Varga, "Detecting DDoS attacks within milliseconds by using FPGA-based hardware acceleration," in *Proceedings of the NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, April 2018.

[43] M. Kuerban, Y. Tian, Q. Yang, Y. Jia, B. Huebert, and D. Poss, "FlowSec: DOS attack mitigation strategy on SDN controller," in *Proceedings of the 2016 IEEE International Conference on Networking, Architecture and Storage (NAS)*, August 2016.

[44] P. Zhang, H. Wang, C. Hu, and C. Lin, "On denial of service attacks in software defined networks," *IEEE Network*, vol. 30, no. 6, pp. 28–33, 2016.

[45] Y. Jia, F. Zhong, A. Alrawais, B. Gong, and X. Cheng, "Flowguard: an intelligent edge defense mechanism against IoT DDoS attacks," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9552–9562, 2020.

[46] M. P. Novaes, L. F. Carvalho, J. Lloret, and M. L. Proenca, "Long short-term memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment," *IEEE Access*, vol. 8, pp. 83765–83781, 2020.

[47] M. Odusami, S. Misra, O. O. A. Alli, and A. A. Adebayo, "A survey and meta-analysis of application-layer distributed denial-of-service attack," *International Journal of Communication Systems*, vol. 33, no. 18, Article ID e4603, 2020.

[48] org and m. ininet, 2021, http://mininet.org/.

[49] inveceorg>, "S. S. a.a. Kali tools," 2021, https://tools.kali.org/information-gathering/hping3.

[50] Inc, Netdata for Single-Node Monitoring, https://www.netdata.cloud/agent/.