Self-similarity-based DDoS Detection for Software-defined Networks

Mohamad Khattar Awad
Computer Engineering Department
Kuwait University
Alshadadiyah, Kuwait
mohamad@ieee.org

Ghazal Alsholi
Computer Engineering Department
Kuwait University
Alshadadiyah, Kuwait
ghazal.alsholi@ku.edu.kw

Haniah Altabaa
Computer Engineering Department
Kuwait University
Alshadadiyah, Kuwait
haniah.altabaa@ku.edu.kw

Dania Hani Abu Daqar
Computer Engineering Department
Kuwait University
Alshadadiyah, Kuwait
dania.abudaqar@ku.edu.kw

Shahad Alshaher
Computer Engineering Department
Kuwait University
Alshadadiyah, Kuwait
s.alshaher@ku.edu.kw

Hamed M.K. Alazemi
Computer Engineering Department
Kuwait University
Alshadadiyah, Kuwait
hamed@eng.kuniv.edu.kw

***Abstract*—The emergence of software-defined networks (SDN) has made the network flexible, programmable, and responsive to change. However, the architectural characteristics of SDN make the network vulnerable to various security attacks. Distributed denial of service (DDoS) is one such attack that aims to exhaust the network's computational and bandwidth resources, blocking legitimate users' access. This paper aims to study the effectiveness of using the self-similarity property, measured by the Hurst-exponent, in detecting DDoS attacks on the SDN controller. Simulation results have shown that normal and attack traffic flows captured at the controller have different degrees of self-similarity. Specifically, attack traffic flows show high levels of self-similarity, whereas normal traffic flows show lower degrees. Experiments demonstrated an average of 35% difference between the Hurst-exponent values of normal and attack traffic flows obtained, making it effective in classifying traffic flows and detecting attacks.**

***Keywords*—Software Defined Networks, DDoS Mitigation, Self-Similarity, Saturation Attacks, Hurst-exponent, Security**

I. Introduction

Legacy networks were operated on hardware-centric devices, requiring intensive human intervention to configure and maintain the network performance [1]. Due to changing business and customer requirements, networks must be constantly modified and reconfigured to meet the changing requirements. The emergence of Software Defined Networking (SDN) architecture made networks flexible, programmable, scalable, and responsive to change, as it decoupled the logic from the data forwarding units. In particular, it decouples the control plane and the data plane. The control plane makes decisions and controls the process of packet forwarding. Whereas the data plane devices forward traffic according to predefined rules set by the controller [2], revolutionizing networking by allowing for centralized control. OpenFlow is a communication protocol commonly used in SDN. It allows a central controller to control network devices directly [3].

The architectural characteristics of SDN have introduced security challenges, making the network vulnerable to various security attacks [4, 5]. Distributed Denial of service (DDoS) attacks are crucial attacks that aim to exhaust the computational and bandwidth resources of the network. DDoS attacks have become the leading threat to SDN since they target control, data planes, switches, and communication channels. When the switch receives a packet, it matches it with one of the rules in its flow table. If the packet does not match any existing rules, the switch encapsulates it and passes it to the controller to make a decision. After analyzing the packet and identifying its route, the controller sends a new rule to the switch to update its flow table. DDoS exploits this scenario and tries to send a large number of excessive packets that do not match the existing flow table rules. In this case, the communication channel between the switch and the controller is kept busy for a long time. Moreover, the computational resources in the controller, including the CPU and memory, will be exhausted, and the flow tables will be overflowed with new rules related to illegitimate users [6].

DDoS attacks can cause serious business disruptions and financial losses by making services inaccessible to users. Attacks on critical applications such as government and military systems can lead to significant implications and losses. This motivates researchers to develop secure and efficient ways to secure SDN networks against such attacks.

In this study, the self-similarity property measured by the Hurst-exponent[1] is used to detect different DDoS attacks. The Hurst-exponent is a mathematical tool used in time series analysis to predict trends for various applications. It can be used to find whether the data follows a random walk or has persistent or anti-persistent behavior. In networks, one method to detect or predict saturation attacks is using the

[1]The Hurst-exponent is a statistical method for assessing the long-term memory of time series data. It measures the likelihood of a time series continuing its current trend or reverting to its average value over time[7].

979-8-3315-2096-0/25/$31.00 ©2025 IEEE

Hurst-exponent and calculating the degree of self-similarity [8]. There are many methods to estimate the Hurst-exponent, including the Rescaled range (R/S) analysis method, which was used in [8], the Detrended Fluctuation Analysis (DFA), the Wavelet Transform, and the Variogram method. The study in [8] reported that regular OpenFlow traffic has low self-similarity. However, it has a higher self-similarity (a Hurst-exponent closer to 1) during saturation attacks. This property was useful in detecting saturation attacks.

The rest of this paper is organized as follows. An extensive review of the literature on attack detection and mitigation is provided in Section II. The proposed solution and methodology are presented in Section III. The results are analyzed and discussed in section IV. Conclusions summarize the key findings are presented in section V.

II. Literature Review

In this section, we present a review of the existing literature on detecting and mitigating SDN attacks.

Authors in [9] proposed SDNGard, a dual-stage defense system to reduce DoS control plane saturation attacks. The approach consists of two phases, detection and mitigation, which are done by adding a data plane to store packet cache temporarily and a probabilistic packet migration module to manage traffic flow between the control plane, data plane, and the cache during an attack. The detection system works by adding a simple global flood counter for each node triggered and incremented for every packet that triggers a miss. The system moves to a defense stage when a counter exceeds a predefined threshold. The mitigation model starts with initiating a probability of sending packets directly to the controller rather than the cache. This probability then decreases as the system detects more abnormal activity. The data plane cache slows packet processing to protect the controller from attack overload. This approach showed a high rate of legit packet delivery when under attack. However, its detection mechanism, which relies on a simple counter, may lead to misinterpretation. The approach can only reduce the attack and cannot wholly prevent it. Moreover, it introduces hardware costs and network complexity concerns, making evaluating its cost-effectiveness necessary.

Swami et al. [2] developed a new intrusion detection system (IDS) that employs a machine learning classifier to classify the TCP-SYN flooding. Five different machine learning classification models were used, and performance was evaluated. Due to the lack of SDN-related datasets, a new dataset was created by collecting traffic data at the target host. The proposed IDS consists mainly of three modules. The first module generates normal and attack traffic, extracting features and converting them into a format that can be fed to the classifier. In contrast, the second module handles data pre-processing. Finally, the third module invokes the classifier to verify whether the traffic belongs to a normal or attack class. Nonetheless, since the network features constantly vary, it becomes difficult to classify and detect attacks using ML efficiently.

Furthermore, Ahuja et al. [10] proposed a hybrid ML model that concatenates support-vector classifier with Random Forest to classify the traffic and detect attacks. Moreover, they created a dataset by extracting 23 significant features affecting detection accuracy. The proposed technique depends on predefined thresholds for the number of packets in the switch and the controller. The dataset is created during the monitoring interval, where the flow and port statistics at each switch are analyzed, and the data is converted into a CSV file to feed to the ML model.

Other studies have analyzed some of the statistical properties of the traffic and exploited them to detect attacks. Z. Li et al. [8] have proposed a new statistical-based mechanism called SA-Detector for anomaly detection by exploiting statistical characteristics of the traffic between the controller and the switches in the data plane. Specifically, the degree of self-similarity has been used to detect malicious traffic, and it was measured using the Hurst-exponent. The approach was tested against 63 attacks generated by all possible combinations of 6 main attacks. The Hurst-exponent was estimated using the rescaled range method based on the network flow's byte size and packet count. When the controller receives many Packet-In messages, the traffic is redirected to the OpenFlow switch cache, which will temporarily install the table-miss packets to avoid overloading the network resources. After that, a calculation component analyzes the packets stored in the cache and calculates the Hurst-exponent to determine whether the traffic is normal or malicious. If malicious traffic is detected, new flow rules will be installed in the corresponding vSwitch flow table to block the malicious hosts. The SA detector proved to be efficient in detecting saturation attacks.

Some of the ways to defend against or mitigate DoS saturation attacks include analyzing network flow using statistical approaches, implementing rate limits to reduce the controller's load from specific sources, load balancing to evenly distribute traffic, firewall, access control policies to restrict unauthorized access, network monitoring tools to detect anomalies promptly, and segment the network into smaller subnetworks to reduce the impact of an attack, all while maintaining regular firmware and software updates [11]. In this study, a Hurst statistical-based model will be developed to detect DDoS attacks in SDN.

III. Materials and Methods

This section outlines the methodology used in this study to detect DDoS attacks using the Hurst-exponent. The study evaluates the effectiveness and accuracy of using the Hurst-exponent as an attack detection mechanism for SDN openflow environment networks. This setup creates an SDN network by integrating a central controller with a Mininet simulation environment. Operating on an OpenFlow protocol, a Ryu controller is used to manage and control the network. For experimentation purposes, results were obtained from a custom Mininet topology consisting of four switches and eight hosts, as shown in Fig. 1; other topologies were also experimented with to ensure consistency.

This section describes the consecutive stages used in the study to develop the proposed model. The stages are as follows: Dataset Generation, Packet Analysis, and finally, the computation of the Hurst-exponent.

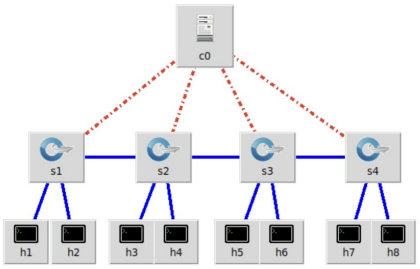

Fig. 1. Mininet Topology

A. Dataset generation

Normal and attack-type network traffic have different characteristics. The Hping3 function generates traffic with varying characteristics in the Mininet environment for simulation purposes.

Normal network traffic flow and DDoS attack-type traffic flow were captured on two different nodes: the switch and the controller. The traffic on the switch was captured every 1s, whereas the traffic on the controller was captured on three different time scales: 100ms, 10ms, and 1ms. SYN, UDP, and ICMP DDoS flooding attacks were simulated in this study by initiating attacks with random fake IP addresses to analyze the change in the Hurst-exponent under different attack conditions. On the other hand, the normal traffic was simulated using combinations of SYN, UDP and ICMP with an average packet arrival rate of about 2 packets per second.

B. Packet analysis

Packets in the datasets are analyzed based on fields that provide significant classification impact, such as the byte size of a packet and the number of packets within the time interval [8]. The Hurst-exponent is used to evaluate the self-similarity of the network based on these two variables. Hence, each flow type generates two Hurst-exponent values: one for the packet size and the other for the number of packets.

C. Hurst-exponent calculation

The Hurt-exponent measures the randomness in the time series and its long-term persistence and patterns. The values of the Hurst exponent are bounded between 0 and 1. A Hurst value of less than 0.5 implies that the data interchanges between high and low values. However, a value greater than 0.5 indicates persistence, while a value around 0.5 indicates full randomness in the time series. Several methods can be used for Hurst exponent calculation; however, this study uses the rescaled range (R/S) method introduced in [8]. A Python function was used to estimate the Hurst-exponent: Hurst exponent is calculated by generating a list of window sizes with a minimum window size of 8 values. This value was chosen as it produced optimal results for the used topology. Then, for each window size, the time series is divided into small segments accordingly to calculate R/S for each segment. The arithmetic mean of the R/S values for each window size is obtained across all segments. A log-log plot of window sizes versus the R/S values is finally used to estimate the plot slope, which is the Hurst-exponent, by performing linear regression. The calculation is performed according to equations 1 to 7 [12], where X_ij represents the is the j^{th} value in the i^{th} segment of the time series.

- Calculate the mean value

$$\bar{X}_i = \frac{1}{L}\sum_{j=1}^{L} X_{i,j} \tag{1}$$

- Construct a mean adjusted series

$$Y_{i,j} = X_{i,j} - \bar{X}_i \tag{2}$$

- Construct a cumulative deviate series

$$Z_{i,j} = \sum_{k=1}^{j} Y_{i,k} \tag{3}$$

- Calculate the range of values series

$$R_i = \max(Z_{i,j}) - \min(Z_{i,j}) \qquad \forall i \tag{4}$$

- Calculate the standard deviation series S_i:

$$S_i = \sqrt{\frac{1}{L}\sum_{j=1}^{L}(Y_{i,j})^2} \tag{5}$$

- Calculate the rescaled range series

$$\left(\frac{R}{S}\right)(L) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{R}{S}\right)_i, \tag{6}$$

- where

$$\left(\frac{R}{S}\right)_i = \frac{R_i}{S_i} \tag{7}$$

IV. Results and Evaluation

This section presents the findings from our experiments and analysis of the results. The results were evaluated by finding the averages and the medians and plotting the Box-and-Whisker plots. Furthermore, K-fold cross-validation methods were used for evaluation. The results demonstrated a difference between the Hurst-exponent of the OpenFlow normal and attack traffic due to the self-similarity nature of SDN flood attacks. A Hurst-exponent value was compared by evaluating and calculating the values at different points in the SDN environment for both attack and normal OpenFlow traffic scenarios.

Figures 3- 5 show the packet counts in the normal and abnormal flows per time unit for 100 ms and 1 ms time scales, respectively. Figures 3 and 2 show random behavior of normal traffic, indicating an expected Hurst value of around 0.5. However, abnormal traffic has persistent behavior indicating a Hurst exponent greater than 0.5, as shown in Figures 4 - 5.

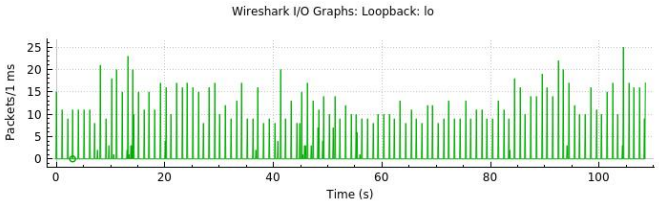

Fig. 2. Normal 1ms Time-Scale

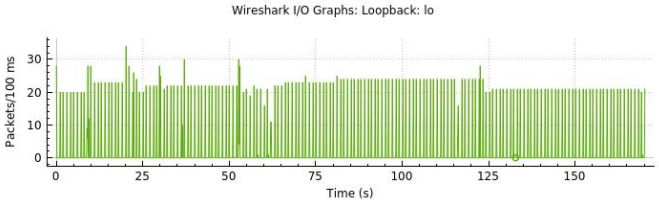

Fig. 3. Normal 100ms Time-Scale

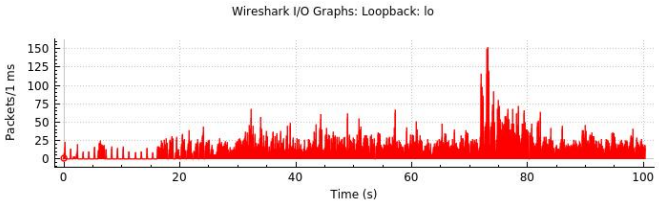

Fig. 4. Attack 1ms Time-Scale

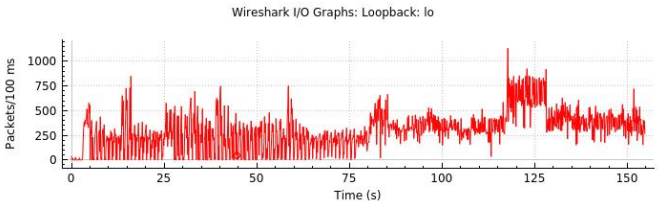

Fig. 5. Attack 100ms Time-Scale

TABLE I
COMPARISON OF HURST-EXPONENT VALUES OF TRAFFIC TYPES AT DIFFERENT COLLECTION TIME-SCALES

Collection Time-Scale	100ms	10ms	1ms
H-exp. for Normal Traffic	0.45336	0.41886	0.40476
H-exp. for Attack Traffic	0.8249	0.813725	0.80694
Avg. Difference	0.37154	0.394865	0.40218

The results in TABLE. I show the average values of the Hurst-exponent for the traffic collected at the controller under normal and attack conditions with different time scales. It can be seen that normal traffic shows inherent randomness since the average values of all time scales are around 0.5. On the other hand, the average values of the Hurst-exponent for the attack traffic are greater than 0.7, indicating persistence in the data. Furthermore, the table demonstrates that the difference between the normal and attack Hurst exponent values is around 0.37, allowing for effective traffic classification.

TABLE II
AVERAGE HURST-EXPONENT TRAFFIC ANALYSIS ON A SWITCH

Flow-Type	Hurst exponent
Normal Traffic	0.8894
Attack Traffic	0.91166

Conversely, TABLE. II presents the average values for the Hurst-exponent for the normal and attack flows on the switches. The difference between the two scenarios is extremely small, with a Hurst exponent difference of less than 0.1. This small difference makes it challenging and insufficient for classifying and detecting attacks. This result is consistent with the study in [12], where it was observed that SDN hosts had Similar Hurst-exponent values for both normal and attack traffic.

TABLE III
ATTACK-TRAFFIC AVERAGE NUMBER OF PACKETS AND NUMBER OF BYTES FOR DIFFERENT SCALES

	Packets			Bytes		
Scale	100ms	10ms	1ms	100ms	10ms	1ms
Average H-exp.	0.8249	0.813725	0.80694	0.87505	0.82965	0.82925

The byte size of the network flow also demonstrated a similar self-similarity behavior as in the packet count, as shown in Table III. The table summarizes the Average Hurst exponent values for flood attacks for three different time scales.

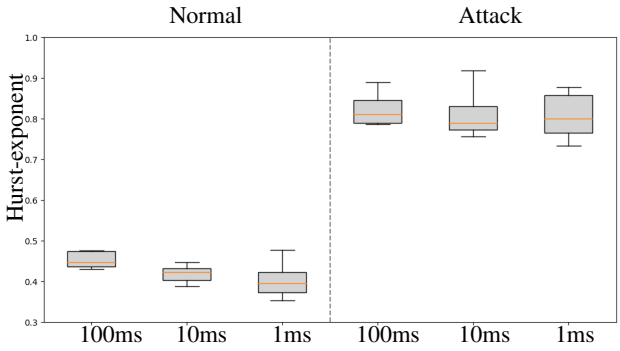

Fig. 6. Hurst-exponent of controller traffic for normal and attack traffic.

Fig. 6 and 7 show Hurst-exponent values under normal and attack conditions for the controller and the switch, respectively. Experimental data illustrates that the Hurst-exponent of the controller under attack is substantially higher than that of the normal condition. On the contrary, it is noticeable that the extent of the change in the Hurst-exponent for the switch under normal and attack conditions is smaller than that of the controller. Therefore, the indicator used in this paper for identifying whether the traffic is normal or an attack is the Hurst-exponent, computed for the controller traffic, which successfully detects attacks aimed at the controller.

Fig. 8 shows the Hurst-exponent of 10-fold cross-validation for normal and attack OpenFlow traffic collected at the controller, respectively. The K-fold cross-validation method

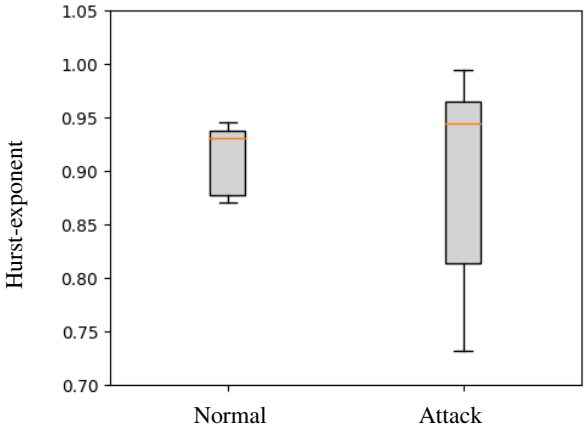

Fig. 7. Hurst-exponent of switch traffic for normal and attack traffic.

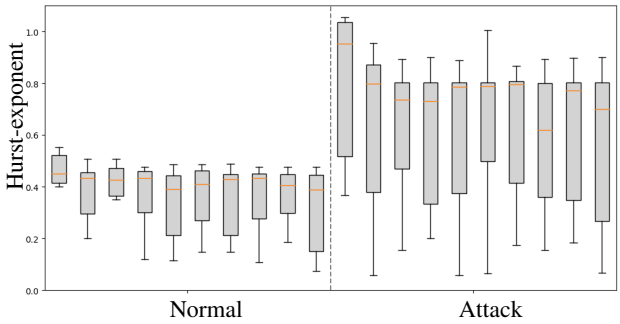

Fig. 8. Hurst-exponent of Controller Traffic for Normal and Attack Traffic

examines the Hurst-exponent across different SDN datasets with different time series lengths for normal and attack traffic flows. Each dataset is split up into 10 folds with different lengths. Ten iterations were performed over various traffic flow time-series obtained from the controller for each traffic type to ensure the robustness of the results. The normal traffic flow has a Hurst-exponent median of around 0.45, while the attack flow has a Hurst-exponent median of around 0.8. The results show significant differences between the Hurst-exponent values of attack and normal traffic flows. Specifically, the attack Hurst-exponent consistently indicates higher values than the normal one. This behavior persisted among different time series lengths, highlighting the accuracy and effectiveness of our findings. The results suggest that the Hurst exponent is an effective metric for detecting attack traffic flows.

It's worth noting that the highest obtained Hurst exponent value for the normal flow is less than 0.5. At the same time, the flood attack traffic had low Hurst exponent values, with the Hurst exponent being as small as 0.067, creating false negative attack detection possibilities.

V. Conclusion and Future Work

The Hurst exponent is a statistical method that measures a time series's long-term memory and self-similarity. This paper examined the effectiveness of self-similarity analysis in detecting DDoS attacks in the SDN environment. The experimentation was conducted on a mininet-based environment simulating both normal and flood network traffic. Results demonstrated a persistent network-flow behavior during flood attacks, resulting in a higher Hurst exponent. Furthermore, normal traffic had an anti-persistent behavior with a smaller Hurst exponent. The attack and normal OpenFlow traffic had an average result difference greater than 35%. These results align with previous literature studies, suggesting that statistical methods such as the Hurst-exponent can detect malicious DDoS traffic in SDN environments.

This work's results provide a promising approach to enhancing SDN security. Future research should focus on further analyzing traffic flow collection methods. In addition, it should investigate the use of ML to identify high-impact traffic types and dynamically allocate the Hurst exponent attributes, such as the window size.

VI. Acknowledgment

This work was supported and funded by Kuwait University Research Grant number EO03/24.

References

[1] N. Anerousis, P. Chemouil, A. A. Lazar, N. Mihai, and S. B. Weinstein, "The Origin and Evolution of Open Programmable Networks and SDN," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1956–1971, 2021, conference Name: IEEE Communications Surveys & Tutorials.

[2] R. Swami, M. Dave, and V. Ranga, "Detection and Analysis of TCP-SYN DDoS Attack in Software-Defined Networking," *Wireless Personal Communications*, vol. 118, no. 4, pp. 2295–2317, Jun. 2021.

[3] A. Lara, A. Kolasani, and B. Ramamurthy, "Network Innovation using OpenFlow: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 493–512, 2014, conference Name: IEEE Communications Surveys & Tutorials.

[4] D. Tang, Y. Yan, C. Gao, W. Liang, and W. Jin, "LtRFT: Mitigate the Low-Rate Data Plane DDoS Attack With Learning-To-Rank Enabled Flow Tables," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 3143–3157, 2023, conference Name: IEEE Transactions on Information Forensics and Security.

[5] N. Shahzad, G. Mujtaba, and M. Elahi, "Benefits, Security and Issues in Software Defined Networking (SDN)," *NUST Journal of Engineering Sciences*, vol. 8, no. 1, pp. 38–43, 2015, number: 1.

[6] S. Kaur, K. Kumar, and N. Aggarwal, "Analysis of DDoS Attacks in Software Defined Networking," in *2022 IEEE Delhi Section Conference (DELCON)*, Feb. 2022, pp. 1–6.

[7] J. Beran, Y. Feng, S. Ghosh, and R. Kulik, *Long-Memory Processes: Probabilistic Properties and Statistical Methods*. Berlin, Heidelberg: Springer, 2013.

[8] Z. Li, W. Xing, S. Khamaiseh, and D. Xu, "Detecting Saturation Attacks Based on Self-Similarity of OpenFlow Traffic," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 607–621, Mar. 2020, conference Name: IEEE Transactions on Network and Service Management.

[9] J. S. Maddu, S. Tripathy, and S. K. Nayak, "SDNGuard: An Extension in Software Defined Network to Defend DoS Attack," in *2019 IEEE Region 10 Symposium (TENSYMP)*. Kolkata, India: IEEE, Jun. 2019, pp. 44–49.

[10] N. Ahuja, G. Singal, D. Mukhopadhyay, and N. Kumar, "Automated DDOS attack detection in software defined networking," *Journal of Network and Computer Applications*, vol. 187, p. 103108, Aug. 2021.

[11] Z. A. Bhuiyan, S. Islam, M. M. Islam, A. B. M. A. Ullah, F. Naz, and M. S. Rahman, "On the (in)Security of the Control Plane of SDN Architecture: A Survey," *IEEE Access*, vol. 11, pp. 91 550–91 582, 2023.

[12] H. Lan and Y. Pan, "Sdn abnormal traffic detection algorithm based on rescaled range analysis," in *2019 Computing, Communications and IoT Applications (ComComAp)*. IEEE, 2019, pp. 231–236.