International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

Enhancing DDoS Attack Detection in Software-Defined Networks with Entropy-Based Techniques

Adam Gorine[1], Mohamed Abdelrahman[2]
[1]School of Computing and Creative Technology, UWE, Bristol, UK
[2]School of Computing and Creative Technology, UWE, Bristol, UK

Abstract— *The introduction of Software-Defined Networks (SDN) represents significant advancements in network design by separating control and forwarding planes. While SDN improves network administration productivity, it has many vulnerabilities which hackers can exploit. One such cyber-attack is Distributed Denial of Service (DDoS), which leads to many challenges. This paper aims to assess SDN vulnerabilities by using a novel technique, Entropy, that can detect DDoS attacks at an early stage. The methodology relies on Entropy to identify abnormal network behaviour, which may indicate DDoS attacks. In addition, a novel mitigation technique using flow drop rules enables the rapid and targeted suppression of malicious traffic. Therefore, it enhances the security of SDN network devices. The solution implements a three-stage DDoS attack detection system for the SDN environment. It involves data gathering, entropy calculation, and threshold-based detection to identify potential attacks.*

Keywords— *Software-defined networks (SDN), DDoS Attacks, Entropy-Based Detection, Flow Drop Rules, Network Security, Threat detection.*

I. INTRODUCTION

Software-defined networks (SDN) are a revolutionary technique for simplifying network administration by separating the data and control planes.

The introduction of SDN has allowed it to program the network control layer. As a result, it is possible to disconnect the physical network hardware from the control plane by moving the control logic from networking devices such as switches and routers in traditional networks to a centralised unit as the controller.

Designing and implementing new protocols is easier because of the separation between the physical network and the controller. Examples of network services include access control, quality of service (QoS), enforcing new rules, bandwidth management, and traffic engineering.

Although the concept of SDN might seem complicated with the separation of data and control plane, it is simple in execution. It will store Flow entries in tables on the data plane.

A secure transport layer protocol provides secure communication with a controller about new entries not in the flow table. Each flow entry includes matching rules and actions so the data plane knows what to do when a flow matches another one. Each packet's header fields are checked against the flow table's matching rules. If a match is discovered, the switch will execute the action stated in that flow entry; otherwise, it will transmit the packet to its controller for further processing. A flow rule is then created and put in the switch flow tables along the selected route by the controller after it processes the header.

Numerous security issues might arise due to the controller's centralised nature, as it acts as a single point of failure in the network, as demonstrated in reference [27]. The network is entirely down by taking out the controller, making the attacks much more effective.

A distributed denial of service (DDoS) attack might cause the controller to become unresponsive. DDoS attacks include sending many packets to one or more hosts to overwhelm the target host's ability to respond. If the source addresses of incoming packets are spoofed, which almost are, the switch will not be able to find a match and will have to forward the packet to the controller instead. Aggregating valid and DDoS-faked packets can tie the controller's resources into a state of continuous processing, which may exhaust them and render the controller inaccessible for packets that have just arrived. It may even cause the controller to go down, resulting in the loss of the SDN architecture.

Because of the centralised nature of the controller in SDN, it is an ideal target for attackers to exploit. Because the attack packets are transmitted with many fake source IP addresses, the DDoS attack could create difficulties for both switches and the controller. Each time an attack packet arrives, a new flow rule must be formed, and as a result, the switch must record the packets in its memory and pass the header data to the controller to ensure the attack is successful. Receiving many malicious packets will use a significant amount of switch memory. It will ultimately result in the installation of several additional flow entries in the switch's flow tables. This may result in the exhaustion of memory and a significant slowdown of the flow table lookup, and in the worst-case situation, it may result in the switch being brought down. Meanwhile, these packets from across the network will be routed to the controller for further processing and analysis. Moreover, the enormous number of packets delivered to the controller, each consuming a portion of the controller's memory and processing capacity, may ultimately cause the controller to fail, as elaborated in reference [1].

DDoS attacks are more effective in SDN networks and do more damage than in traditional networks. Because of this, it is critical to have an efficient and trustworthy detection technique in place to identify such attacks. In big data centres with multiple switches, it is vital to discover the targeted network sections via the detection process. This will reduce the time it takes to implement a mitigation strategy. Switches have far less memory and power than controllers, making

Adam Gorine, and Mohamed Abdelrahman, "Enhancing DDoS Attack Detection in Software-Defined Networks with Entropy-Based Techniques," *International Research Journal of Advanced Engineering and Science*, Volume 9, Issue 2, pp. 45-53, 2024.

them more vulnerable to failures. This makes the switches more vulnerable to these sorts of attacks. Therefore, it is essential to have fast emergency techniques in place to protect the switches from breaking down as soon as the first signals of an attack are detected.

This research paper aims to develop a lightweight detection and mitigation system against DDoS attack threats using a simulation of Software-defined Networks in Mininet.

II. Background on SDN

2.1 Software-Defined Networks (SDN)

SDNs are a new network design and administration method that uses software instead of hardware. In traditional networking, even slight changes in network circumstances would require large-scale reconfiguration of switches, routers, and other networking equipment. However, this approach is much more straightforward and flexible in SDN [2].

Figure 1 illustrates the interplay between a typical network node's data and control planes. The control plane is responsible for determining the network pathways and pushing them to the data plane so that data may be forwarded to its destination. It is thus necessary to adjust the settings of each device to change the flow policy once it has been created. Network operators may have to spend significant time re-configuring devices regularly to keep pace with evolving bandwidth demands and network capacity in extensive networks, as illustrated in [3].

As in the conventional network, the control plane is no longer spread across network nodes but rather centralised at the controller, which interacts with network nodes to set up the data plane over a southbound SDN protocol, as mentioned in [4]. OpenFlow is a southbound SDN protocol that enables controllers and network nodes to communicate.

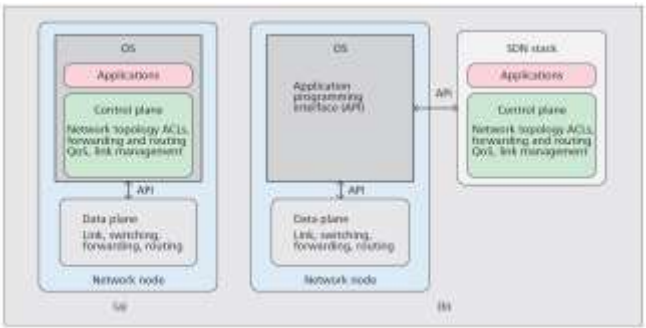

Fig. 1. SDN Data and Control Planes [3].

2.2 SDN Controllers

SDN controllers are the network's brain at the heart of SDN network architecture. OpenFlow switches are connected using the southbound APIs, whereas SDN applications interact with the northbound APIs, as discussed in [5] (Bui and Aberkane, 2016).

The controller employs different modules to execute various tasks, such as identifying network devices and acquiring network information. Controller add-ons can be incorporated to strengthen and broaden the controller's capabilities, such as network monitoring and detecting traffic anomalies.

The POX controller is a popular choice among researchers and academics. POX's lightweight OpenFlow controller may benefit SDN research, teaching, and experimentation. With a simple and lightweight architecture, POX has been used to develop SDN projects and conduct academic research. POX is used to implement the suggested algorithm in this research project, although this does not imply that the approach is tied to a specific controller. Mininet, a virtualised network, will make the setup easier and more reusable [1].

2.3 SDN Security

Attacks on well-known firms, banks, government agencies, and colleges are becoming more common, and data may be stolen, exposed, or altered in a malicious attack. Sadly, the SDN has several issues and weaknesses that make it an attractive target for criminals. In SDN, a wide range of threats might be identified. These SDN threats and OpenFlow issues will be discussed in the following sections.

2.3.1 SDN Threat Vectors

According to the authors in [6], there are seven types of threat vectors in SDN, as illustrated in Figure 2. Table 1 outlines various risks to SDN and traditional networks, some specific to SDN while others affect both. The initial threat vector aims to flood switches by generating legitimate-looking traffic flows potentially caused by a switch failure. This method can potentially launch Denial of Service (DoS) attacks on networking equipment, including switches, constituting the second method of attack. The primary cause of this vulnerability lies in weaknesses within switches, allowing attackers to disrupt the network by redirecting traffic flows to other switches, consequently impeding network traffic.

Switches in Software-Defined Networking (SDN) systems can be vulnerable to various attacks due to inherent weaknesses or misconfigurations. Some vulnerabilities of switches include Flow table Exhaustion, OpenFlow protocol vulnerability and misconfiguration of switches as illustrated in reference [26]

The third danger vector is an assault on the communications between the control and data planes. Several ways to exploit this vulnerability include forging traffic flows and overloading the controller. As a result, DDoS attacks are directed towards the controller. TLS/SSL protocol flaws are the primary cause of this attack, as identified in [7]. Vulnerabilities in the controller are exploited in the fourth vector of attack against the control plane. The whole network might be brought down if an attack on the control plane is successful. Lack of communication between the management plane and controller is the sixth danger.

Attackers may employ the third, fourth, and fifth threat vectors to launch a DDoS attack and knock down the whole system.

Attacks against administrative stations are the sixth danger vector. A lack of reliable resources for cleanup and investigation adds to the seventh danger vector. Using these tools, you can quickly, securely, and correctly recover from an attack, as pointed out in [8].

Adam Gorine, and Mohamed Abdelrahman, “Enhancing DDoS Attack Detection in Software-Defined Networks with Entropy-Based Techniques,” *International Research Journal of Advanced Engineering and Science*, Volume 9, Issue 2, pp. 45-53, 2024.

International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

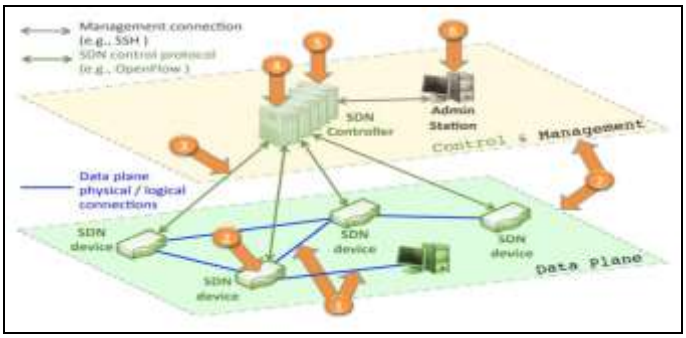

Fig. 2. The seven threat vectors of SDN [8]

TABLE 1. SDN Specific vs non-specific threats [8]

Threats	SDN Specific	Risk to SDN
Vector 1	No	This threat can lead to DoS attacks with increased potential impact.
Vector 2	No	The impact is potentially increased.
Vector 3	Yes	Communication with centralised controllers can be exploited
Vector 4	Yes	Hacking the controller could compromise the entire network.
Vector 5	Yes	Malicious applications can now be easily created and deployed on the controller.
Vector 6	No	The impact is potentially increased.
Vector 7	No	It is still crucial to ensure fast recovery and diagnosis when faults occur.

2.3.2 Denial of Service (DDoS) Attacks

Numerous security challenges afflict network security, with denial of service (DoS) attacks a prominent concern. Global instances across various cases and scenarios highlight the widespread impact of such attacks. Additionally, the centralised nature of Software-Defined Networks (SDNs) is recognised as a vulnerability, making them particularly susceptible to Distributed Denial of Service (DDoS) attacks.

The primary goal of such attacks is to prevent network services from functioning. This might be accomplished by using all the network bandwidth or consuming the memory and CPU on the service provider nodes. To overwhelm the service, the attacker sends large numbers of packets to suck up all of the available bandwidth or takes over the whole processing capacity of the service provider nodes. Sending a large number of UDP packets, for example, might be used to produce the attacking traffic and prevent regular communication from reaching its target [9]. The complexity of attack prevention, detection, and mitigation must be increased as these attacks constantly evolve, and hackers are always searching for new ways to develop DDoS attacks.

In Distributed Denial of Service (DDoS) attacks, attackers often use compromised sources to send numerous packet streams. A proficient attacker may manipulate packet fields and traffic characteristics to evade detection solely based on traffic categorisation rules. This manipulation aims to deceive defence mechanisms, making these altered packets appear as regular traffic, as mentioned in reference [9].

2.3.3 DDoS Attacks Operation

The attacker identifies vulnerable computers or nodes that may be accessed over the internet. The attacker has to ensure that the hosts they choose can execute the application they want to use. An attacker may take control of the host computer by exploiting any vulnerabilities that have been discovered. These scripts are meant to utilise a portion of their host system's resources so that their owners would not be able to notice any significant concerns about the system's performance. Conversely, the code is meant to be concealed in the best possible manner to evade detection by the host system's programs. Zombies must be in contact with each other and the attacker to find out which hosts may participate in the attack or to utilise them to update the Host's code. UDP, TCP, and ICMP protocols are all used to communicate with each Host, which a handler manages. The attacker sets up the attack, including the target victim's IP address, the attack traffic's port, length, TTL, traffic type, and other details. All of these properties may be changed during the attack phase to evade detection, as referenced in 11].

III. LITERATURE REVIEW

This section highlights past and current research done on DDoS attack detection and mitigation techniques in Software Defined Networks. It shed light on various techniques, from hardware-based defences to entropy-based detection systems, highlighting the new research on the security and resilience of SDNs against new cyber threats. Despite the progress made to date, current techniques have challenges and limitations. Therefore, more research needs to be done on the security of SDNs.

The researchers in reference [12] presented a hardware-based defensive system in SDN architecture to combat HTTP GET Flooding attacks by implementing a per-URL counting technique on an FPGA-based defensive system and extending the NetFPGA-based OpenFlow switch. This FPGA is a technique to distinguish the network traffic based on the differences between regular users' and bots' behaviours.

In a separate study by [13], researchers proposed a detection method for detecting DDoS attacks using the Sequential Probability Ratio Test (SPRT). They classified flow events connected with an interface for precise detection, demonstrating faster response times, increased options and improved accuracy compared to previous FPGA-based detection techniques.

Researchers [14] suggested countermeasures against UDP flooding by monitoring the quantity of incoming or outgoing packets. They used the controller to distinguish normal and malicious traffic based on packet quantity, enabling the detection of a high number of packets immediately.

Their work [15] utilised the maximum entropy value and flow-based traffic attributes, such as source and destination IP addresses and ports. They developed a comprehensive system for detecting and mitigating DDoS attacks, worm propagation, and port scan attacks.

The study also involved comparing CPU and flow table size usage between two data collecting methods, native OpenFlow and sFlow, on real network traffic data. For instance, adopting sFlow helped reduce the number of factors that needed calculation, resulting in decreased communication between switches and controllers. This reduction contributed to a lower false-positive rate for their technique.

Adam Gorine, and Mohamed Abdelrahman, "Enhancing DDoS Attack Detection in Software-Defined Networks with Entropy-Based Techniques," *International Research Journal of Advanced Engineering and Science*, Volume 9, Issue 2, pp. 45-53, 2024.

The primary mitigation strategy described in reference [16] included implementing drop actions and higher-priority bidirectional flow rules, effectively reducing the attack surface. However, it's important to note that their experimental setup featured only one switch, and the emphasis on the deployment location for mitigation was limited. Using a predetermined threshold value, the researchers successfully distinguished between legitimate and malicious communications. It's worth mentioning that the method's accuracy may be affected by the sample flow rate.

In reference [17], the authors proposed that Entropy-based DDoS attacks were run locally at each local edge switch using a modified version of Open vSwitch in a mininet emulator and a FloodLight controller to decrease communication costs. They tested the system using the CAIDA dataset and compared detection and false-positive rates at various monitoring intervals. Flow rules with a drop action were installed at the source switch of the attack to drop the attacked packets. Their study found that shorter monitoring intervals led to a faster detection time. They highlighted the importance of detecting attacks on the networks directly linked to local edge switches.

In another research paper [18], the authors developed anomaly detection on SOHO and ISP networks using well-established techniques such as maximum Entropy, Network Traffic Anomaly Detection (NETAD), Rate-Limiting and Threshold Random Walk with Credit-Based Rate Limiting (TRW-CB). They found that the home networks' accuracy was superior to Internet Service Providers but lacked explicit mitigation methods.

Joint-entropy-based DDoS detection using several packet attributes was suggested by [19]. They used flow time, packet length, source address, and destination port as the main factors to identify distinct forms of DoS and DDoS attacks. In addition, they ran tests on virtual campus networks built on top of SDN architecture.

A practical and lightweight framework for detecting and mitigating DDoS attacks in SDN has been suggested by [20]. They first gathered network traffic data using the SDN controller and sFlow agents. Then, they adopted an entropy-based approach to quantify network characteristics. They developed a Support Vector Machine (SVM) classifier for timely and accurate attack detection, coupled with mitigation mechanisms based on blacklists and traffic relocation.

Authors in [21] proposed SAFETY as an innovative approach for detecting and preventing TCP SYN flooding. This method combines programming with the visibility of Software-Defined Networking (SDN). It also includes an entropy technique to assess the unpredictability of flow data. The entropy data consists of the IP address of the destination and a subset of TCP flags, effectively enhancing the identification and mitigation of TCP SYN flooding.

The research conducted by [22] recommended using Fast Entropy algorithms by the SDN controller to detect DDoS attacks. Real-time prevention of DDoS attacks was achieved by harnessing SDN capabilities in conjunction with the Fast Entropy approach. This technique employs SDN and the Fast Entropy algorithm to collect and analyse data efficiently, detect DDoS incidents, block malicious packets, and redirect legitimate flows to the designated destination.

DDoS attacks may be detected in SDN environments utilising the entropy measure and the changes in host role profiles for detecting under-attack states, according to [23]. In addition, they approached the problem of time while gathering data. They applied a statistical approach to evaluate flow data supplied by OpenFlow switches, identifying early-stage DDoS attacks.

Entropy-based algorithms, while capable of identifying DDoS attacks, have notable limitations. One significant constraint is the calculation of the probability distribution of a feature using a single value. Although this method proves useful for data analysis, it results in the loss of the distribution of the examined characteristic. Consequently, in certain circumstances, the anomalous effects might be obscured. Also, it could not distinguish between distinct distributions with the same degree of uncertainty as this technique. Since there is no randomisation in malicious communication, it will go undetected by the system.

IV. METHODOLODY

This detection system is based on two fundamental principles: the Entropy fluctuation of the destination IP address and the traffic Flow Rate. Using Entropy will provide us with a lightweight and effective solution for SDN architecture with a single controller that can be implemented quickly and effectively simultaneously.

4.1 Entropy-Based DDoS Detection

Entropy-based DDoS Detection is a cybersecurity technique that uses entropy analysis, focusing on destination IP addresses and their duplication frequency, to identify and mitigate Distributed Denial of Service (DDoS) attacks. The method involves calculating entropy values, establishing a baseline under normal conditions, and triggering alerts or countermeasures when the computed Entropy consistently deviates from the baseline, indicating potential DDoS threats.

4.1.1 Entropy Calculation

Entropy, also known as the Shannon-Wiener index, is a fundamental concept in information theory. In this context, it quantifies the unpredictability or randomness of a random variable, specifically the final destination IP. The entropy range is [0, $\log_2 m$], where 'm' represents the number of destination IP addresses (equal to 1 when m=1). In scenarios like a DDoS attack, where all traffic converges on a single destination, network entropy reaches its minimum. On the contrary, Entropy reaches its maximum when viable destinations are distributed evenly [24].

The detection mechanism relies on entropy analysis, as described by [25]. Entropy analysis involves using fixed-size windows to collect data, with window size determined by packet count or elapsed time. The window size, determined by the quantity of packets transmitted within a designated time frame, is used to group packets based on their destination IP addresses.

Adam Gorine, and Mohamed Abdelrahman, “Enhancing DDoS Attack Detection in Software-Defined Networks with Entropy-Based Techniques,” *International Research Journal of Advanced Engineering and Science*, Volume 9, Issue 2, pp. 45-53, 2024.

The characteristic metric is the destination IP address, and randomness is gauged by the frequency of different destination IP addresses within the window. The relative frequency $\mathbf{F_i}$ for each IP address $\mathbf{IP_i}$ is calculated using Equation below:

$$F_i = \frac{n_i}{n}$$

Where n_i is the number of packets with destination IP address IP_i and n is the total number of packets.
This formula computes the entropy H based on the relative frequencies of different destination IP addresses within the observed packet window. Which is calculated as follows:

$$H = -\sum_{i=1}^{m} F_i \log_2 F_i$$

$$0 \leq F_i \leq 1 \; implies \; H \geq 0$$

Entropy reaches its maximum when the relative frequencies of all **'m'** IP addresses are equal. For example, in a scenario with 50 packets, each having a unique destination and a computed probability (F_i=1/50), the Entropy is 5.643. However, if 10 out of the 50 packets flow to a single destination address, the Entropy decreases to 5.213.

In a normally functioning network, increased packets directed to a single host or a small group result in decreased Entropy, indicating unusual behaviour. An observed decrease in Entropy during an attack is an early detection signal. SDN networks, particularly susceptible to DDoS attacks, necessitate swift detection. The detection window encompasses fifty packets to achieve a trade-off between speed and computational load [25]. For entropy calculations, a new module in the pox controller collects fifty packets, corresponding to a window of fifty flow start requests. Based on flow start rates, a timer determines the collection time utilised in the subsequent detection step. The controller calculates the shortest path for each flow but doesn't save computed pathways by default. A function tracks calculated pathways to aid in determining attack routes."

The entropy function, utilising destination IP addresses and their duplication frequency, calculates Entropy (Ec). Under normal to low traffic conditions, a default entropy value sets the initial entropy threshold value (Eth). If the computed Entropy (Ec) consistently falls below the threshold for five consecutive instances, it raises suspicion of an attack, triggering the need for additional analysis.

4.1.2 Selecting a threshold.

Given the window size set at 50 and the assumption of 50 or more connected hosts, selecting an appropriate threshold for DDoS detection is critical to entropy-based detection. In the new function, each set of 50 Packets in messages is parsed for their destination IP addresses, and the Entropy of the list is calculated for each group. This estimated Entropy is then compared to a predefined threshold. A cyberattack is deemed to have occurred if the estimated Entropy has been below the threshold for at least five consecutive entropy periods. With 250 attack packets, this corresponds to a detection rate of 5 entropy periods, providing the network with an early warning. Experimentation with values from one to five successive periods revealed that using five periods yielded the lowest number of false negatives, positives and the lowest false positive rate. In a window of 50 packets with a network of 50 or more hosts, maximum Entropy occurs when each of the 50 packets is evenly distributed across all hosts. During an attack, the significant increase in packets directed to the same destination host or subnet renders the target inaccessible to legitimate traffic for an extended period, aligning with the attack's primary goal. The assault involves directing packets towards a single host or subnet. As long as the assault rate on a host exceeds the regular traffic level, which is typically the case, the number of packets sent to that Host within a given time period increases, leading to a decrease in Entropy by a certain percentage. Falling below the threshold is considered an indication of an ongoing assault.

4.1.3 Mitigating through Flow Drop Rules

The algorithm must implement attack mitigation measures if a switch is identified as being under attack. Possible approaches include installing flows in the attack paths to drop packets until the attack ceases or blocking the incoming ports from which the attack traffic is arriving.

While these methods effectively mitigate the attack and provide time for network operators to identify the attack sources before the controller or switches break down, their adoption also impacts legitimate traffic. Legitimate network services may become unavailable or respond slowly due to the measures taken.

Controllers are typically designed with high capacities, ensuring they do not crash rapidly. However, switches have limited resources and are less resilient against attacks. During an ongoing attack, the flow table on switches may fill with a large number of short flows, eventually leading to switch failure. To safeguard the integrity of network switches, the selected strategy involves implementing port blocking and packet dropping as a mitigation measure.

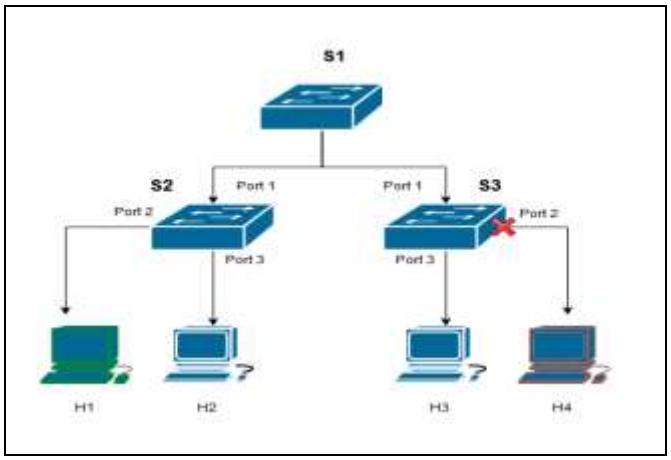

Fig. 3. Mitigation by dropping packets.

Consider the network topology depicted in Figure 3, which consists of three switches and four hosts forming a tree structure. Let's consider a scenario where host 4 (highlighted in red) initiates an attack on host 1 (highlighted in green). Our

Adam Gorine, and Mohamed Abdelrahman, "Enhancing DDoS Attack Detection in Software-Defined Networks with Entropy-Based Techniques," *International Research Journal of Advanced Engineering and Science*, Volume 9, Issue 2, pp. 45-53, 2024.

strategy entails blocking all packets from the infected Host at the closest point to the attack source—specifically, port 2 on switch 3.

4.2 Proposed Mitigation Algorithm

The algorithm presented as a flowchart in Figure 4 illustrates the proposed detection and mitigation method.

Our work process can be divided into three main functions: calculation, detection, and mitigation.

In the calculation phase, we will gather data from incoming traffic around the network and count the rate of every destination IP address. We will compute their entropy values by grouping every 50 packets within the session. Then, we will compare the entropy value against the pre-set threshold of 1.26. The entropy value should consistently stay below the threshold for five consecutive periods to detect a potential network attack.

Upon attack detection, the system will pull the data from our calculations to identify the attack path and block the port connected to the infected Host.

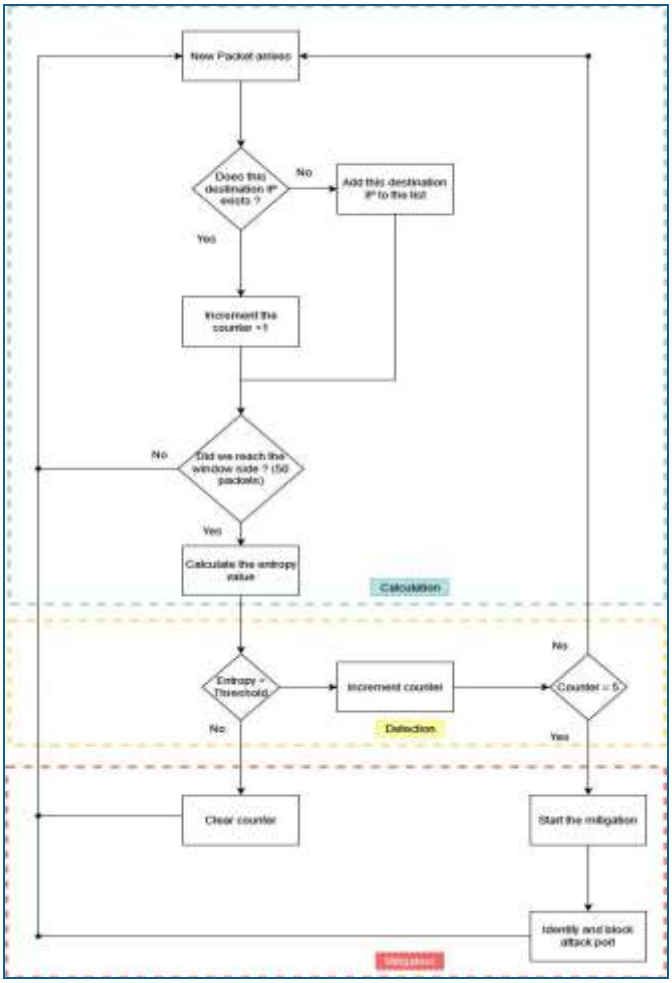

Fig. 4. Proposed Solution-Flowchart

V. EXPERIMENTAL SETUP

In this setup, we use Mininet to simulate a virtualised network environment, providing a flexible and scalable platform for conducting experiments on DDoS attacks and mitigation in software-defined networks. The design of the network topology is carefully planned to reflect the specific experimental scenarios we intend to investigate.

Our network topology is structured as a tree, featuring two levels of depth and eight fanouts, consisting of nine switches and 64 hosts, as depicted in Figure 5. Mininet serves as the network emulator in this experiment, simulating a real network and functioning as an industry-standard tool for SDN implementation.

The initial step in our testing is selecting a controller. POX was the preferred controller in our testing as it is a well-known controller valued for its speed, lightweight nature, and compatibility with Linux, Mac OS X, and Windows, and it has topology discovery capabilities.

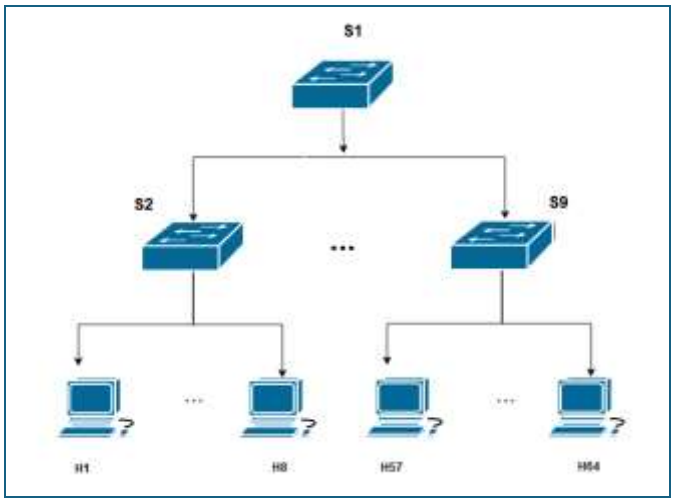

Fig. 5. Mininet Topology

5.1 Test Scenarios and Results

To have a solid test for our solution, we must test the system under two different types of DDoS attacks:

1. Concentrated DDoS Attacks: In this scenario, all the attack traffic is directed towards a single host.
2. Scattered DDoS Attacks: This involves distributing the attack traffic among multiple hosts.

This comprehensive testing approach allows us to test the entropy calculation when the attack is scattered, and it will be harder to detect as the value depends on randomness. Additionally, we will vary the attack rate in each test run, ranging from 15% to 75%, for both concentrated and scattered attack types.

The experiment contains five different DDoS attacks. Initially, three different attack rates target a single host. Subsequently, we launch two different attack rates on four hosts connected to the same switch and subnet.

Throughout the experiment, regular traffic is maintained across all switches, with packets randomly generated and sent to all hosts.

A script manually triggers the execution of attacks, explicitly initiating them after one-fourth of the simulation's duration.

Adam Gorine, and Mohamed Abdelrahman, "Enhancing DDoS Attack Detection in Software-Defined Networks with Entropy-Based Techniques," *International Research Journal of Advanced Engineering and Science*, Volume 9, Issue 2, pp. 45-53, 2024.

International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

Throughout a Mininet session, IP addresses are systematically assigned to all hosts, starting from 10.0.0.1. For a single-host attack, we randomly chose a host in a switch and directed it to send attack packets to another host. In contrast, the remaining hosts and switches continued normal operations. In the case of a four-host attack, a randomly selected host transmits attack packets while the rest of the network functions without disruption.

In the three test cases involving a single-host attack, the attack rates were set at 25%, 50%, and 75% of the attack traffic to normal traffic ratio. Throughout the entire testing process, two Scapy applications operated in the background—one generating regular traffic and the other initiating an attack that accelerated packet delivery. This unintentionally led to a scenario where between 9 and 14 packets out of 50 were sent to the same destination IP address during the 25% rate attack.

This unplanned outcome was associated with occasional minor issues in Mininet. To remedy this, the attack rates were adjusted to augment the number of attack packets in the other two scenarios. At the 50% rate, 26 out of 50 packets were directed to the same destination IP address, and at the 75% rate, 39 out of 50 packets faced a similar targeting.

Subnet attacks will be executed at two distinct attack rates: 50% and 75%. Notably, the 25% attack rate is not employed in subnet attacks. In the case of four hosts, the 25% attack rate results in an average of 12 attack packets, distributing three packets to each Host. This rate is considered standard for the controller and does not pose a significant threat.

We established a subnet comprising four hosts to assess an attack on a group of hosts. In a 50% rate attack, each Host received between 5 and 7 packets; in a 75% rate attack, each Host obtained between 9 and 10 packets. Although the variation in attack packet numbers was unintentional, it inadvertently increased the proportion of attack traffic in the test, decreasing the entropy value. For instance, in a 25% rate attack on a single host, the attack packets ranged from 9 to 14, accounting for 18% to 28% of the total attack packets sent.

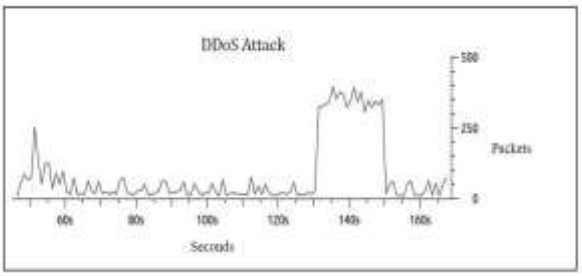

Fig. 6. DDoS packet count against regular traffic

In practical terms, DDoS attacks exhibit significantly higher intensity than what is apparent on the surface. Most often, attacks generate traffic that is many times greater than typical traffic patterns. For example, a standard attack might generate 250 packets per second, whereas regular traffic only produces 50 packets per second (refer to Figure 6). If such an attack persists in a controller without mitigation, it will exhaust all of the controller's resources in processing the attack packets.

5.2 Attack on One Host

This section examines the impact of an attack on a specific host. Each graph is generated from 10 runs, each involving 4000 packets for the test. However, we focus on 60-packet windows, specifically during the occurrence of the attack. The attack is consistently initiated during the 15th window. On the horizontal axis, each point signifies a window comprising 50 packets, while on the vertical axis, each point represents the Entropy for the corresponding window.

The data presented in the graph represents mean values from ten runs. Figure 7 illustrates the change in Entropy during a 25% rate attack. The blue line corresponds to regular traffic across all graphs, while the red line depicts the Entropy shift during the launch of attack traffic. Figure 7 delineates the contrast between the Entropy values for regular and abnormal traffic.

The first six entropies in the graph are consistently lower than our threshold of 1.26. The lowest point in the confidence interval for regular traffic is 1.24, and the highest point for attack entropy is 1.36. In the initial test with a 25% rate attack, no attack was detected due to the disparity between our suggested threshold and the entropy value. While three values were below the threshold, the absence of five consecutive values prevented the detection of the attack.

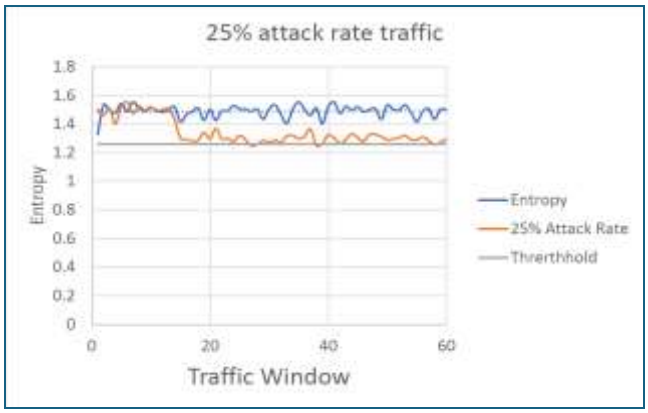

Fig. 7. 25% Rate Attack on One Host

In Figure 8, the results of our approach become evident. Our findings reveal the effectiveness of detecting any attack that consumes 50% or more of the incoming bandwidth when directed at a single site. The simulation was conducted ten times with a 50% success rate, enabling us to calculate the success rate based on these ten iterations. Notably, no false negatives were detected, even when an attack was ongoing but went unnoticed by the controller. This underscores the capability of detecting DDoS attacks within the first 250 arriving packets with a 100% success rate. This success rate holds true across all other instances without any undetected attacks.

Two higher-rate tests were conducted on the same Host to examine more concentrated attacks, as shown in Figures 13 and 14. Figure 8 illustrates an attack with a 50% success rate, while Figure 9 depicts a 75% success rate attack on a single

Adam Gorine, and Mohamed Abdelrahman, "Enhancing DDoS Attack Detection in Software-Defined Networks with Entropy-Based Techniques," *International Research Journal of Advanced Engineering and Science*, Volume 9, Issue 2, pp. 45-53, 2024.

International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

host. Both simulations are compared against typical traffic volume to illustrate the difference in Entropy between the two scenarios.

As the attack rate increases, the window of opportunity becomes more pronounced and narrower. This results in a significant reduction in the amount of information available to the attacker. The acceleration in the attack rate, combined with a fixed number of attack packets, leads to a higher proportion of attack packets within the window. Consequently, the attack graph becomes more profound and narrower, reflecting the more substantial decrease in Entropy observed in the 75% example.

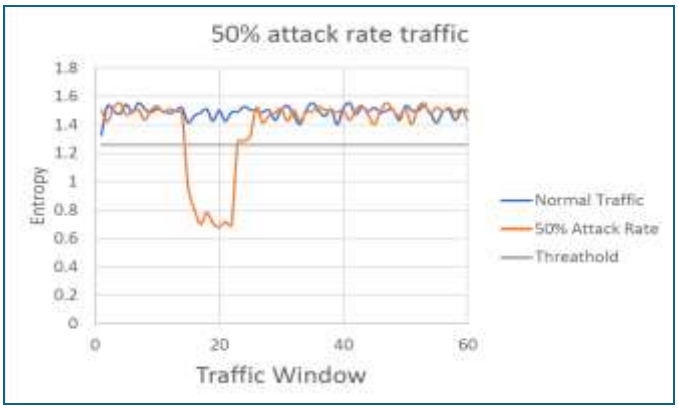

Fig. 8. 50% Attack Rate Traffic

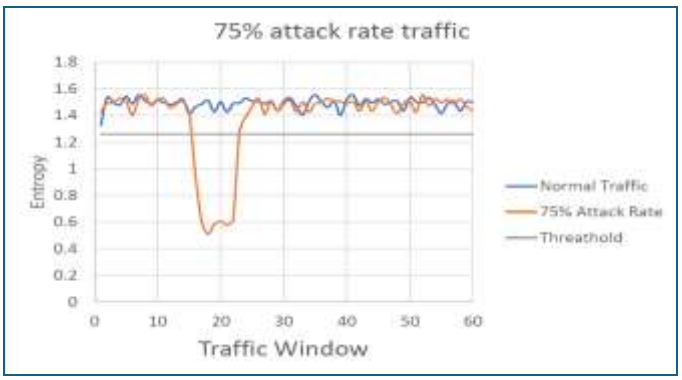

Fig. 9. 75% Rate Attack on One Host

5.3 Attack on Multiple Hosts

In this section, we examined Entropy's effectiveness in detecting attacks on the controller involving four hosts within the same subnet. Given that the baseline for detection was set at a 1.26 rate on one Host, the threshold was maintained at the same level. During ten runs of attacks on a subnet with a 50% rate, the Entropy consistently registered values lower than the threshold. Figure 10 illustrates a decrease in Entropy that is well below the threshold but higher than the 50% rate observed on a single host.

Moving on to Figure 11, it represents a 75% rate attack on four hosts. Notably, there is a sharp drop in Entropy when a substantial number of packets are directed to the same subnet. The confidence interval for both the 75% rate attack on a single host and the 75% rate on a subnet shows the highest confidence interval.

Both tests achieved a 100% success rate in all 20 runs, demonstrating that the system could effectively detect the attack, even when distributed across four hosts. However, it is important to note that the entropy value was higher than the attack on a single host, given that the total number of attack packets targeting each Host had decreased.

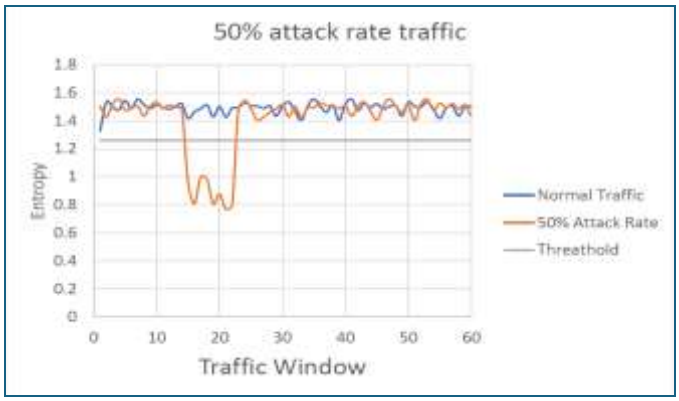

Fig. 10. 50% Rate Attack on Multiple Hosts

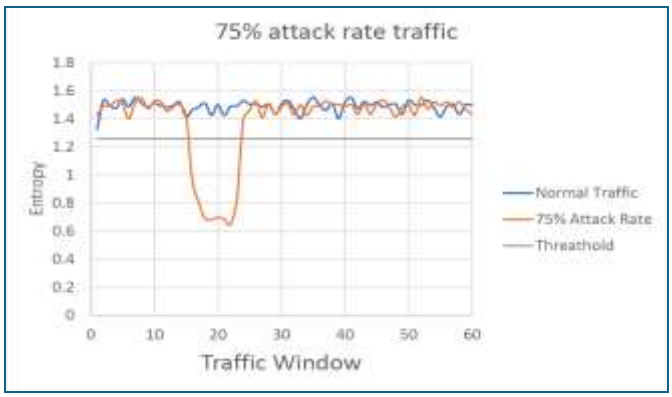

Fig. 11. 75% Rate Attack on Multiple Hots

VI. CONCLUSION

This research aims to provide a reliable and lightweight method for detecting various DDoS attacks during their early propagation phases in SDN networks and suggests a mitigation technique. Unlike attacks on traditional networks, where the goal is often to overwhelm a specific service with excessive traffic, DDoS attacks in SDN can be more dispersed to evade detection while still targeting the controller and switches. To be effective in SDN, a detection mechanism must identify both single and multiple victim attacks with minimal latency to allow for swift mitigation strategies.

Upon integrating our solution into the controller, not only does it recognise malicious activity, but it also identifies the specific attack vectors being targeted. The high detection rates for varied traffic patterns in our data demonstrate that the algorithm can perform effectively across various network settings and is not restricted to a single network scenario.

Entropy was used as a detection tool in this study, and we could identify attacks on a single host or a subnet of hosts in a network. Notably, our technique correctly identified subnet attacks even when the number of packets sent to the controller was as low as 50% of the total traffic. When we adopted a

Adam Gorine, and Mohamed Abdelrahman, "Enhancing DDoS Attack Detection in Software-Defined Networks with Entropy-Based Techniques," *International Research Journal of Advanced Engineering and Science*, Volume 9, Issue 2, pp. 45-53, 2024.

threshold of a 50% rate of attack packets per total traffic, our strategy demonstrated a 100% success rate. In scenarios where DDoS attacks constitute 75% to 100% of all traffic in non-SDN networks, our approach outperformed the nearest technique in identifying the attack.

This paper presents a three-stage DDoS attack detection system designed for SDN environments. During the calculation phase, data is gathered from incoming network traffic, and the rate of each destination IP address is computed. Entropy values are calculated by grouping every 50 packets, and these results are compared against a predefined threshold of 1.26. A potential network attack is identified if the entropy value consistently stays below the threshold for five consecutive periods. Upon detection, the system uses the calculated data to identify the attack path and subsequently blocks the port connected to the infected Host while legitimate traffic continues to flow unimpeded.

REFERENCES

[1] Faizullah, S., Ebadati E., O. M., Zhani, M. F. "*A Comprehensive Survey on Security Challenges and Countermeasures in Software-Defined Networking*." IEEE Communications Surveys & Tutorials, 23(1), pp.577-614, 2021

[2] Mohammed Samaka, Mohammed Abdelkader Khalil, Hossam S. Hassanein, "*Cloud SDN: A Survey on SDN-based Cloud Networking*," IEEE Access, vol. 9, pp. 138677-138704, 2021.

[3] Zhiqiang Zhang, Rongxing Lu, Raymond K. Wong, Xuemin (Sherman) Shen, "Challenges and Solutions of Software-Defined Networking: A Comprehensive Survey," *IEEE Communications*, vol. 23, no. 2, pp. 916-961, 2021.

[4] Zhenyu Li, Haoyang Lu, Jun Li, Guangjie Han, "A Decade of Software-Defined Networking (SDN): A Bibliometric Analysis," *IEEE Access*, vol. 9, pp. 80910-80923, 2021.

[5] Xiaoyu Zhang, Heng Zhang, Liang Zhang, Zhiyuan Guo, Yushu Liu, "Towards a Generic Interface for Programmable Data Plane of Open vSwitch," *IEEE Access*, vol. 9, pp. 126090-126101, 2021.

[6] Chunmei Liu, Jianglong Li, Longxiang Gao, Kai Xing, Zhaoming Lu, "A Comprehensive Survey on Software-Defined Networking: Concepts, Technologies, and Applications," *IEEE Internet of Things Journal*, vol. 8, no. 21, pp. 15933-15955, 2021.

[7] Holz, R., Kiermaier, T., Kammenhuber, N., Carle, G., "X.509 forensics: Detecting and localising the SSL/TLS men-in-the-middle," in *European Symposium on Research in Computer Security*, pp. 217-234, Springer, Berlin, Heidelberg, 2012.

[8] Chunmei Liu, Jianglong Li, Longxiang Gao, Kai Xing, Zhaoming Lu, "A Comprehensive Survey on Software-Defined Networking: Concepts, Technologies, and Applications," *IEEE Internet of Things Journal*, vol. 8, no. 21, pp. 15933-15955, 2021.

[9] Holz, R., Riedmaier, T., Kammenhuber, N., Carle, G., "X.509 forensics: Detecting and localising the SSL/TLS men-in-the-middle," in *European Symposium on Research in Computer Security (ESORICS)*, pp. 217-234, Springer, Berlin, Heidelberg, 2012

[10] Dridi, L., Zhani, M.F., "SDN-guard: DoS attacks mitigation in SDN networks," in *5th IEEE International Conference on Cloud Networking (Cloudnet)*, pp. 212-217, 2016

[11] Huaqiang Yuan, Keqin Li, Jingqiang Lin, Huansheng Ning, "A Survey on DDoS Attacks and Defense Mechanisms in SDN-Based IoT Networks," *IEEE Access*, vol. 9, pp. 40829-40839, 2021.

[12] Jinsoo Park, Eunji Lee, Jongsub Moon, "Mitigation of HTTP Flooding Attacks Using SDN Controller with NetFPGA-Based OpenFlow Switch," *IEEE Access*, vol. 8, pp. 103893-103906, 2020.

[13] Dong, P., Du, X., Zhang, H., Xu, T., "A detection method for a novel DDoS attack against SDN controllers by vast new low-traffic flows," in *IEEE International Conference on Communications (ICC)*, pp. 1-6, 2016.

[14] Tung, Y.H., Wei, H.C., Ti, Y.W., Tsou, Y.T., Saxena, N., Yu, C.M., "Counteracting UDP flooding attacks in SDN," *Electronics*, vol. 9, no. 8, p. 1239, 2020.

[15] Giotis, K., Argyropoulos, C., Androulidakis, G., Kalogeras, D., Maglaris, V., "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," *Computer Networks*, vol. 62, pp. 122-136, 2014.

[16] Hamed HaddadPajouh, Hassan Habibi Gharakheili, Rasool Jalili, Amin Tootoonchian, Albert Y. Zomaya, "sFLow-Assisted Anomaly Detection and Mitigation in Software-Defined Networks," *IEEE Transactions* on Network and Service Management, vol. 17, no. 3, pp. 1555-1569, 2020.

[17] Narges Mehran, Gholamreza Shahrokh Abadi, "An Entropy-Based Distributed Denial of Service Detection Scheme in Software Defined Networks," *Journal of Network and Systems Management*, vol. 29, no. 2, pp. 402-426, 2021.

[18] Mohammed Al-Fayoumi, Adnan Abu-Mahfouz, Mohamed Nabeel, Abderahmane Tamdjid, "A Survey on Anomaly Detection Techniques in Software-Defined Networking," *IEEE Access*, vol. 8, pp. 214146-214167, 2020.

[19] Mao, J., Deng, W., Shen, F., "DDoS flooding attack detection based on joint-entropy with multiple traffic features," in *12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pp. 237-243, 2018.

[20] Hu, D., Hong, P., Chen, Y., "*FADM:* DDoS flooding attack detection and mitigation system in software-defined networking," in *IEEE Global Communications Conference*, pp. 1-7, 2017.

[21] Hemavathy R, Dr. M. S. Rajasree, "Entropy-based Detection and Mitigation of TCP SYN Flood Attack in SDN," in *International Conference on Inventive Computation Technologies (ICICT)*, pp. 98-103. 2021.

[22] Saman Taghavi Zargar, James Joshi, David Tipper, "A Comparative Study of DDoS Attack Detection and Mitigation in SDN and Cloud Computing Environments," *IEEE Access*, vol. 9, pp. 52933-52949, 2021.

[23] Duy, PT, Pham, V.H., "A role-based statistical mechanism for DDoS attack detection in SDN," in *5th NAFOSTED Conference on Information and Computer Science (NICS)*, pp. 177-182. 2018

[24] Li, L., Zhou, J., Xiao, N., "DDoS attack detection algorithms based on entropy computing," in *International Conference on Information and Communications Security*, pp. 452-466, Berlin, 2007.