Entropy-based distributed denial of service attack detection in software-defined networking

Mohammed Ibrahim Kareem[1], Mahdi Nsaif Jasim[2]
[1]Department of Information Networks, Information Technology College, University of Babylon, Al-Hillah, Iraq
[2]Department of Business Information technology, Business Informatics College,
University of Information Technology and Communications, Baghdad, Iraq

Article Info

Article history:

Received Apr 30, 2022
Revised Jun 14, 2022
Accepted Jul 3, 2022

Keywords:

Distributed denial of service
Early detection
Entropy
Python-based software defined networking controller
Software defined networking security

ABSTRACT

Software defined networking (SDN) is a new network architecture that allows for centralized network control. The separation of the data plane from the control plane, which establishes a programmable network environment, is the key breakthrough underpinning SDN. The controller facilitates the deployment of services that specify control policies and delivers these rules to the data plane using a common protocol such as OpenFlow at the control plane. Despite the many advantages of this design, SDN security remains a worry because the aforementioned chapter expands the network's attack surface. In fact, denial of service (DoS) assaults pose a significant threat to SDN settings in a variety of ways, owing to flaws in the data and control layers. This work shows how distributed denial of service (DDoS) attack detection is based on the entropy variation of the destination IP address. The study takes advantage of the OpenFlow protocol's (OFP) flexibility and an OpenFlow controller (POX) to apply the proposed method. An entropy computation to determine the distributed features of DDoS traffic is developed and it is capable of detecting a user datagram protocol (UDP) flood attack after 0.445 seconds this type of attack occurred.

Corresponding Author:

Mohammed Ibrahim Kareem
Department of Information Networks, Information Technology College, University of Babylon
Al-Hillah, Iraq
Email: mohamed.ibrahim@uobabylon.edu.iq

1. INTRODUCTION

Software defined networking (SDN) design separates the control and data planes, and as a result, all of the logic for controlling the network is shifted from the network devices to a logically centralized software-based entity called the controller. The SDN controller is completely up to date on all aspects of the network. It is software that resides on the control plane, and it is used to instruct forwarding behavior to all of the nodes that are scattered across the data plane [1]. On the data plane, nodes are switches that are in charge of managing packet routing according to a schedule created by the network controller. This procedure is performed prior to displaying any of the available redirection options. A fundamental feature of a SDN is that the network controller is always aware of the network's status, and all traffic flows are communicated to the controller at least once during the network's lifetime so that the controller can determine redirection behavior [2]. Every new technology has its own set of pros and limitations. This is no exception with SDN. In the SDN network, management and configuration are handled differently. Effective network security can be achieved with this infrastructure in place. There are two ways to describe this phenomenon. "SDN security" is a term that refers to the capacity to design and enforce network-wide security policies from a central controller's program via the SDN's centralized logical visibility and programmability. "SDN security" refers to the ability to defend or secure the SDN platform, which is vulnerable to a variety of threats; thus, this paper will focus on this aspect, specifically reviewing works that used entropy [3].

Every conceivable model of a network has a limited number of resources. Attacks using distributed denial of service (DDoS) are directed on resources like bandwidth, storage space, and processing power. The purpose of this assault is to exhaust the victim's resources in order to cause disruption to their network infrastructure. In general, distributed denial of service attacks are launched against the network, server, and application resources. In comparison to the conventional network, the SDN uses an entirely unique method for processing packets [4]. In the context of the SDN architecture, the consolidation of all network functions into a single entity (the SDN controller) looks to be somewhat of a double-edged sword. Through the lens of the network topology, it inevitably enhances both the management and operation of the network An attack vector for attackers becomes the SDN controller, which places it at high risk for failure [5]. Reliability and scaling in data centers are hampered by the centralization of the control plane in SDN architecture as shown in Figure 1.

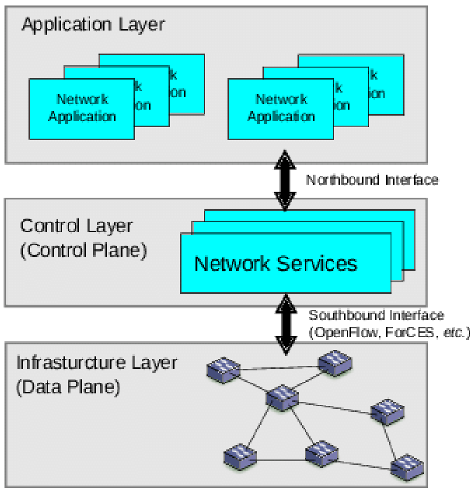

Application Layer
Network Application
Network Application
Northbound Interface
Control Layer (Control Plane)
Network Services
Southbound Interface (OpenFlow, ForCES, etc.)
Infrasturcture Layer (Data Plane)

Figure 1. SDN archetecture: three layers [6]

SDN controllers face three key dangers. In the first attack against SDN-controller communications, the controller and switches are exploited for their lack of security [7]. If the control plane is compromised, the attacker has the ability to launch distributed denial of service attacks. Attacks on controllers' security are a second threat. Because an SDN controller has complete knowledge of the network and can do anything to its own network, any intruder can gain access to the controller and embed malware. This has the potential to destroy the entire network. The third threat is a week-long procedure to atnfirm that the controllers' applications have been properly processed. There are some similarities between this and the previous, but this time the danger is posed by untrusted connections between control-plane applications. user datagram protocol (UDP) flooding DDoS attack overflows a victim with user datagram protocol UDP packets. This attack aims to flood remote server ports with traffic. It prompts the host to search for a snooping application on the port regularly, and if none is found, it sends an internet control message protocol (ICMP) (destination unreachable) packet. This procedure reduces host resources, causing unavailability [8]. A UDP flooding DDoS attack overflows a victim with UDP packets. This attack aims to flood remote server ports with traffic. It prompts the host to search for a snooping application on the port regularly, and if none is found, it sends an ICMP (destination unreachable) packet. This procedure reduces host resources, causing unavailability [9], [10].

Despite the lack of security mechanisms designed to discern malicious intent, researchers are currently focused on finding ways to convey data efficiently from source to destination [11]. These vulnerabilities in security can be used to launch distributed denial of service attacks. A hybrid key, combining classical and quantum key distribution techniques, was proposed by Mahdi and Abdullah [12] to ensure the security of OpenFlow communications in software-defined networks, depending on the quantum's computational complexity and physical features. Using the transport layer security protocol, the hybrid key provides confidentiality, integrity and quantum authentication to secure software-defined network connections between the control and data planes. The current architecture of the internet and networks, as well as their evolution, do not place a strong focus on security [13].

A DDoS attack's internet protocol (IP) packets and traffic can often be distinguished from normal traffic because of their unique characteristics. It is possible to detect DDoS traffic using statistical aspects of IP packet header data such as the entropy of the source IP addresses, or statistical properties of normal and attack traffic patterns. DDoS attack detection strategies have been the subject of numerous studies.

Develop a hybrid approach [14] for detecting denial of service (DoS) attacks and integrate this data into routing decisions so that botnet nodes can be immediately recognized and removed from the network. The suggested solution is sufficiently flexible to for nodes suspected of engaging in a denial-of-service assault to be "rehabilitated" if they halt their harmful activity.

The method proposed by Mousavi and St-Hilaire [15] is works by comparing the entropy of successive packet samples to detect changes in their unpredictability. The entropy is estimated based on the destination IP addresses of a window of 50 packets. An assault is reported if the entropy is less than the threshold. Singh and Jain [16] developed a distributed architecture for analyzing the packet flow behaviour. To identify malicious flows from normal flows, the proposed technique employs entropy and a traceback algorithm. David and Thomas [17], flow-based analysis and a fast entropy technique are used to detect DDoS attacks, according to the authors. The proposed method's detection accuracy has been greatly enhanced. They do a network traffic analysis and calculate the entropy of each flow's request quickly.

The recently published research by Kia [18], the method that is going to be proposed here is going to extend the early detection and mitigation of distributed denial-of-service attacks in software-defined networks, which is based on an Entropy variation of the destination IP address, flow initiation rate, and study of flow specification. These are the three main components of the method.

Saharan *et al.* [19] an effective detection mechanism that is computationally less expensive and capable of detecting various types of attacks with high accuracy is required. As a result, the researchers proposed entropy with dynamic thresholds to detect DDoS attacks in this paper. A dynamic threshold allows us to detect an attack at different rates of traffic with greater accuracy. The CICDDoS-2019 attack dataset was used to validate our approach.

Kalkan *et al*. [20], the authors proposed joint entropy based DDoS defense mechanism scheme in SDN. In their proposed defence scheme, entropy value is calculated for all possible k-element subsets of P where P={IPsrc, IPdst, Psrc, Pdst, Prot, PKTsize, TTL, TCPflag} for $1 \leq k \leq P$. They use score-based packet marking system for filtering the suspicious packets. The method uses a packet-based sliding window with 1000 packets. The method marks the packets as suspicious traffic whenever any current entropy value falls below the threshold value. They use runtime threshold and it is calculated by dividing the sum of current entropy and previous entropy by 2.

Dridi and Zhani [21], DDoS assaults can be prevented by automatically redirecting malicious traffic, adjusting attack flow Timeout settings, and converging flow rules linked to the malicious traffic into an aggregate flow rule set. Controller inbound throughput is reduced by 32%, switch ram by 26%, while packet loss and average package round trip time are both reduced.

Jose *et al*. [22], To counter the surge of DDoS attacks, the authors developed and implemented a machine learning-based attack detection system. Flooding attacks, such as transmission control protocol (TCP) synchronize (SYN), hyper text transfer protocol (HTTP) request flooding, UDP flooding, and ICMP flooding over SDN network traffic.

Current work detects DDoS in software-defined networks. SDN helps developers create network-related applications, such as security and traffic management. The work differs from previous work, which used the POX controller to create an application to detect attacks using early detection based on entropy without any additional tool, rather a modification of the POX controller's l3 learning module. Most of the previous work used Wireshark [23] to analyze networks, which has major flaws that make it hard for the SDN controller to make decisions.

2. PROPOSED DDOS DETECTION SYSTEM

Essentially, it is a form of flood attack. Here, multiple packets are sent to a network device to halt the service or reduce its performance. If the source addresses of inbound packets are faked, the switch cannot detect a match and must forward the packet to the controller. The accumulation of DDoS faked packets and valid packets can bind the controller to continuous processing, exhausting them. This renders the controller inaccessible to newly arriving valid packets. This will cause the controller to fail, resulting in damage to the SDN architecture. For a backup controller, the same difficulty must be met. The proposed method is based on the observation that when a network is under attack, its destination IP frequency varies. Therefore, if we observe a change in this trait, we can recognize an assault. To identify denial of service assaults, we apply two alternative approaches, each of which relies on a unique set of statistical assumptions on the underlying network.

2.1. System architecture

The initial packet is sent from one client to another during normal client-to-client communication in an SDN environment. Initially, there will be no entry in the switch's flow table indicating where the incoming packet should be sent. By default, the packet will be delivered to the controller. The controller will direct the switch to transfer the packet to the proper destination port. In addition, the controller will specify an entry rule for this source and destination in the flow table of the switch.

2.2. POX Modules

The application for detecting will function as independent modules on POX [24]. Using numerous modules, the controller will be able to do several tasks, including installing flows, routing packets, and validating DDoS attacks. The most vital component of the POX controller is the L3 learning module [25].

2.3. Entropy based DDoS detection algorithm

Entropy is used to measure network unpredictability. Randomness increases entropy and vice versa. When entropy falls below a threshold, a DDoS assault has happened. Shannon entropy is key to understanding entropy-based detection algorithms. Entropy measures randomness in a variable [15]. The random variable in our situation is the destination address. The detection approach based on entropy is comparable to that currently in use [26]. Fixed window size is set up so that packets can be collected for entropy analysis. For each window, the packets are put into groups based on the IP address of where they are going. All of these packets that are coming in will have different source addresses. For every new flow, the IP addresses of the destinations will be kept an eye on. These flows that are being watched are put into windows. Each window has a two-column "hash table" or "dictionary." The IP addresses will be listed in the first column, and the number of times it has happened will be listed in the second column. As shown in (1), you can figure out the window equation.

$$H(Sj) = \sum_{i=0}^{N} X_i \tag{1}$$

Where i denotes each one of the window's one-of-a-kind elements and n represents the total number of IP destination addresses for that window's iteration, As shown in (3) can then be used to determine the probabilities associated with each distinct IP destination address.

$$P_i = X_i / \sum_{i=0}^{N} X_i \tag{2}$$

As shown in (2) determines each window. Entropy compares to a threshold. If entropy is over the threshold, no attack has occurred. Any number below the threshold suggests an assault. If no assault is detected, the entropy threshold is lowered to prevent future errors. The detecting system can adapt to network flow in real time. Entropy-based approach detects internal attacks. The attacker and victim are inside the SDN. Attacker can target network controller or clients. In either case, the assault likely has a faked source IP address. Incoming packets don't fit the flow table; therefore, they're sent to the controller. With this approach, the controller may detect if the network is under assault based on the huge quantity of packets with the same destination address. UDP flooding is an example.

To establish an appropriate window size, Oshima *et al.* [26] proposal is based on calculating entropy in small size windows and then using that information to make decisions. Their research concluded that a window width of 50 and 500 successfully detects DoS/DDoS attacks and suggested a window size ranging from 50 to 5,000. This study uses a window size of 50, as suggested by Oshima *et al.* [26]. The OVS's incoming flow is collected by the detection algorithm. There will be a file that keeps track of the number of packets sent and received by each destination IP in the current interval.

2.4. Entropy based detection algorithm

The detection technique is based on three primary inputs: widnows_size, Threshold, entropy estimate. This concept is written in Python and will be implemented as an additional module within the POX controller. In addition, it will include an alerting mechanism. If an attack is identified, an administrator will be notified by short message service (SMS) that the SDN has been compromised. The entropy-based detection algorithm is displayed in the Algorithm 1. The input, which is denoted by the phrase new packet in, refers to a new packet that has arrived with a different source address. Additionally, the destination IP address is investigated to determine whether or not there is already an instance of it in the window. If it is found to be real, then the count associated with that IP address will be increased. In the event that the window is completely occupied, the entropy is calculated, and it is then compared to the threshold. It will be considered an attack if the computed value for an entry is higher than the threshold for five separate counts in a row.

Algorithm 1. Entropy based detection DDoS algorithm
```
Input: Window_Size, Threshold value
Output: Alert message
Begin
While true Do
        Clear hash file
        Capture incoming packets from switch
        Extract relevant features of current packet and store in Hash_File
         // Calculation Entropy
        If number of packets = 50 then
                 Calculate the probability using Eq. 2
 Clear windows size
                 Calculate entropy of network using Eq. 1
         End If
        // DDoS Detection
        If Entropy<= threshold then
                Notify the monitoring server by DDoS detection
        End if
End Do
```

3. SIMULATION AND TEST RESULTS

Mininet is selected as the network emulation platform for the first section of our experiment, which employs POX SDN as the controller. The detecting application will be a separate module integrated within the L3 Learning module. The controller will now be capable of installing flows and verifying DDoS assaults. During this testing phase, OF-switch is used to simulate the behavior of an edge switch in an SDN network. A client in the SDN network generates both regular and attack traffic aimed at a victim node in the Mininet network. A virtual host may be attacked with the Mininet detection mechanism. The IP addresses issued to customers span between 10.0.0.1 and 10.0.0.20. As shown in the Figure 2, two hosts, three open virtual switch (OVS) and a single POX controller are installed for this research.

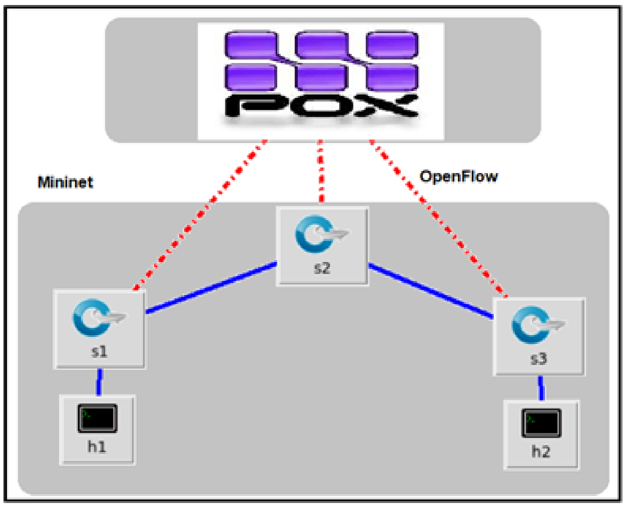
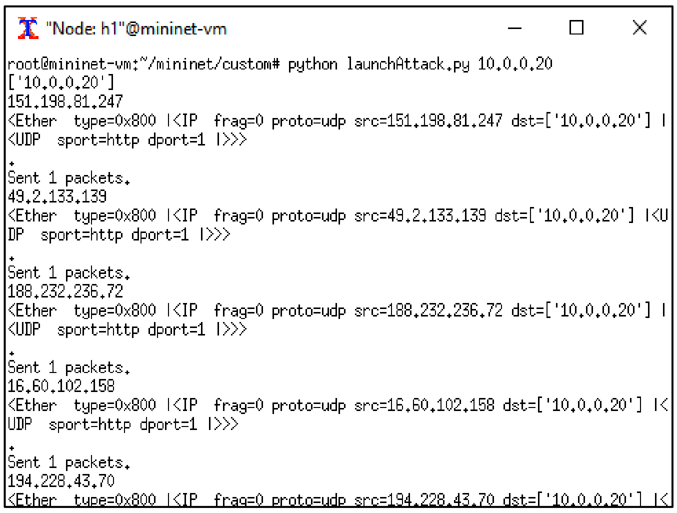


Figure 2. Mininet topology of 3 switches and 2 hosts with single POX controller

Despite the fact that typical networks consist of hundreds of devices, the previous test was designed to simulate comparable problems in large networks due to limited resources. We built a virtual network with the same limitations as our testbed so that we could develop and test our solution in a real-time setting. You can import functions like sendp, IP, UDP, Ethernet, and TCP with the Scapy module [25]. It is used to make UDP packets with different payloads and traffic intervals (inter=0.025) for this work. In steady, normal traffic, it will take 0.1 seconds (inter=0.1) between cars. Figure 3 shows how Scapy script generates traffic in a virtual host (h1).

Simulations and evaluations were carried out on a variety of platforms. The main platform for testing our detection system is a Windows 10 machine with a 2.60 GHz Intel Core i7-6600U processor and 16 GB RAM. There are two virtual machines in this environment (Mininet and POX controller). Before launching an attack with Scapy, launch normal traffic with Scapy to a predetermined threshold value that serves as a boundary between normal and attack traffic (DDoS attack). After several testing, the average entropy for a normal traffic is calculated using normal traffic simulation.

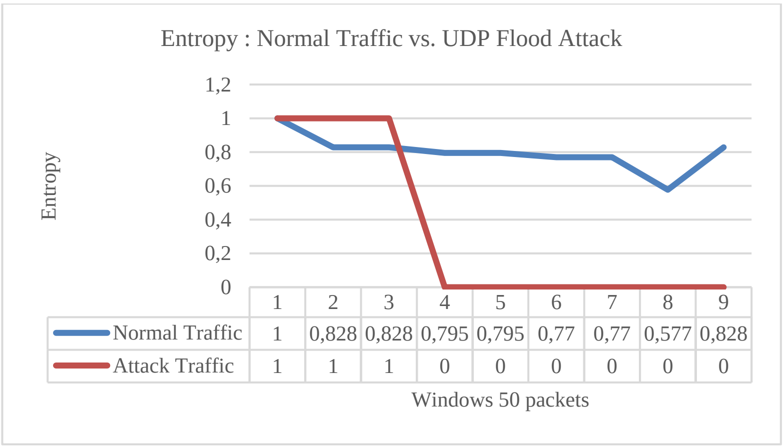


```
root@mininet-vm:~/mininet/custom# python launchAttack.py 10.0.0.20
['10.0.0.20']
151.198.81.247
<Ether  type=0x800 |<IP  frag=0 proto=udp src=151.198.81.247 dst=['10.0.0.20'] |
<UDP  sport=http dport=1 |>>>
.
Sent 1 packets.
49.2.133.139
<Ether  type=0x800 |<IP  frag=0 proto=udp src=49.2.133.139 dst=['10.0.0.20'] |<U
DP  sport=http dport=1 |>>>
.
Sent 1 packets.
188.232.236.72
<Ether  type=0x800 |<IP  frag=0 proto=udp src=188.232.236.72 dst=['10.0.0.20'] |
<UDP  sport=http dport=1 |>>>
.
Sent 1 packets.
16.60.102.158
<Ether  type=0x800 |<IP  frag=0 proto=udp src=16.60.102.158 dst=['10.0.0.20'] |<
UDP  sport=http dport=1 |>>>
.
Sent 1 packets.
194.228.43.70
<Ether  type=0x800 |<IP  frag=0 proto=udp src=194.228.43.70 dst=['10.0.0.20'] |<
```

Figure 3. Traffic generation using scapy

In contrast to attack packets, which has short time flows with tiny amounts of traffic, normal traffic is described as traffic with lengthy duration flows. The purpose is to calculate the average normal entropy value for a two-node network. On the network, a traffic with a 0.5-second traffic interval was begun. (1/0.5=2 packets/sec) defines the traffic rate. The graph in Figure 4 shows how the entropy changes during traffic. During this traffic flow, the lowest threshold was 0.828, and the highest point was 1 in first traffic. This small network's average entropy value is 0.577.

The topology and parameters used to mimic attack traffic with numerous victims are the same as those used to simulate normal traffic. The traffic timing and pace are the only differences. We tested a 25% attack rate for the multiple victim attack. Figure 4 depicts the assault traffic interval: normal traffic and a UDP flood attack on entropy.



	1	2	3	4	5	6	7	8	9
Normal Traffic	1	0,828	0,828	0,795	0,795	0,77	0,77	0,577	0,828
Attack Traffic	1	1	1	0	0	0	0	0	0

Figure 4. Entropy of normal traffic and UDP flood attack

The 2 packets/sec generated for lawful traffic in this multiple victim attack scenario. The attack was discovered, as expected. The developed SDN app, as expected, detects the attack in less than one second. The assault is identified at 0.445 second with a higher traffic rate of 28 packets/sec. As can be seen in Figure 3, the entropy values for both test cases are nearly identical. The entropy levels for the 25% attack rate are slightly greater, ranging from 0.511 to 0.00.

A DDoS attack's average detection rate is between 0.445 and 0.501 seconds. It's safe to state that the algorithm does a fantastic job of discriminating between normal and malicious data. It's vital to keep in mind that this simulation was created expressly for the test. The network and its traffic will grow in a real-world setting. If this is the case, the threshold must be changed to avoid false positive detection.

4. CONCLUSION

A UDP flooding attack is a DDoS attack that floods a target with user datagram protocol (UDP) packets. The goal of this attack is to flood a remote server's arbitrary ports with traffic. As a result, the host is periodically prompted to check for an application snooping on that port, and if none is found, an ICMP (destination unreachable) packet is sent back. In addition, this procedure could lead to the host becoming unavailable. A DDoS attack that overwhelms a victim with UDP packets is known as a UDP flooding assault. The goal of this attack is to flood arbitrary ports on a remote server with traffic. If no snooping application is found, the host will respond by sending out an ICMP (destination unreachable) packet to notify the user that the port has been closed. As a final consequence, this procedure uses up more of the host's resources, putting it at risk of going down.

REFERENCES

[1] K. Kirkpatrick, “Software-defined networking,” *Communications of the ACM*, vol. 56, no. 9, pp. 16–19, 2013.

[2] U. Rahamathullah and E. Karthikeyan, “Distributed denial of service attacks prevention, detection and mitigation–A review,” in *Proceedings of the International Conference on Smart Data Intelligence (ICSMDI 2021)*, 2021, p. 16.

[3] M. I. Kareem and M. N. Jasim, “The current trends of DDoS detection in SDN environment,” in 2021 *2nd Information Technology To Enhance e-learning and Other Application (IT-ELA)*, 2022, pp. 29–34, doi: 10.1109/it-ela52201.2021.9773744.

[4] Q. Yan, Q. Gong, and F. A. Deng, “Detection of DDoS attacks against wireless SDN controllers based on the fuzzy synthetic evaluation decision-making model,” *Ad-Hoc and Sensor Wireless Networks*, vol. 33, no. 1–4, pp. 275–299, 2016.

[5] M. A. Aladaileh, M. Anbar, I. H. Hasbullah, Y. W. Chong, and Y. K. Sanjalawe, “Detection techniques of distributed denial of service attacks on software-defined networking controller-a review,” *IEEE Access*, vol. 8, pp. 143985–143995, 2020, doi: 10.1109/ACCESS.2020.3013998.

[6] W. Braun and M. Menth, “Software-defined networking using OpenFlow: protocols, applications and architectural design choices,” *Future Internet*, vol. 6, no. 2, pp. 302–336, May 2014, doi: 10.3390/fi6020302.

[7] D. Kreutz, F. M. V. Ramos, and P. Verissimo, “Towards secure and dependable software-defined networks,” in *HotSDN 2013-Proceedings of the 2013 ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, 2013, pp. 55–60, doi: 10.1145/2491185.2491199.

[8] K. F. Hassan and M. E. Manaa, “Detection and mitigation of DDoS attacks in internet of things using a fog computing hybrid approach,” *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 3, 2022.

[9] X. Hu, X. Chen, W. Liu, and G. Dai, “Road traffic status prediction approach based on K-means-decision tree model,” *Journal of Engineering, Project, and Production Management*, vol. 12, no. 2, pp. 108–115, May 2022, doi: 10.32738/jeppm-2022-0010.

[10] R. Yaegashi, D. Hisano, and Y. Nakayama, “Light-Weight DDoS mitigation at network edge with limited resources,” in *2021 IEEE 18th Annual Consumer Communications and Networking Conference, CCNC 2021*, 2021, pp. 1–6, doi: 10.1109/CCNC49032.2021.9369635.

[11] J. Mirkovic and P. Reiher, “A taxonomy of DDoS attack and DDoS defense mechanisms,*” Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004, doi: 10.1145/997150.997156.

[12] S. S. Mahdi and A. A. Abdullah, “Improved security of SDN based on hybrid quantum key distribution protocol,” in *Proceedings of the 2nd 2022 International Conference on Computer Science and Software Engineering, CSASE 2022*, 2022, pp. 36–40, doi: 10.1109/CSASE51777.2022.9759635.

[13] B. Chinnaiah, “Protection of DDoS Attacks at the Application Layer: HyperLogLog (HLL) Cardinality Estimation,” in *Cognitive Informatics and Soft Computing*, Springer, 2021, pp. 595–604.

[14] N. M. AbdelAzim, S. F. Fahmy, M. A. Sobh, and A. M. B. Eldin, “A hybrid entropy-based DoS attacks detection system for software defined networks (SDN): A proposed trust mechanism,” *Egyptian Informatics Journal*, vol. 22, no. 1, pp. 85–90, 2021, doi: 10.1016/j.eij.2020.04.005.

[15] S. M. Mousavi and M. St-Hilaire, “Early detection of DDoS attacks against software defined network controllers,” in *Journal of Network and Systems Management*, 2018, vol. 26, no. 3, pp. 573–591, doi: 10.1007/s10922-017-9432-1.

[16] S. Singh and S. Jain, “A review of detection of DDoS attack using entropy-based approach,” *International Journal of Computer Science and Technology*, vol. 4, no. 2, 2013.

[17] J. David and C. Thomas, “DDoS attack detection using fast entropy approach on flow-based network traffic,” *Procedia Computer Science*, vol. 50, pp. 30–36, 2015, doi: 10.1016/j.procs.2015.04.007.

[18] M. Kia, “Early detection and mitigation of DDoS attacks in software defined networks,” Thesis, Ryerson University, 2021.

[19] S. Saharan, V. Gupta, N. Vora, and M. Maheshwari, “Detection of distributed denial of service attacks using entropy on sliding window with dynamic threshold,” in *Lecture Notes in Networks and Systems*, 2022, vol. 449 LNNS, pp. 424–434, doi: 10.1007/978-3-030-99584-3_37.

[20] K. Kalkan, L. Altay, G. Gür, and F. Alagöz, “JESS: joint entropy-based DDoS defense scheme in SDN,” *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2358–2372, 2018, doi: 10.1109/JSAC.2018.2869997.

[21] L. Dridi and M. F. Zhani, “SDN-Guard: DoS attacks mitigation in SDN Networks,” in *Proceedings - 2016 5th IEEE International Conference on Cloud Networking, CloudNet 2016*, 2016, pp. 212–217, doi: 10.1109/CloudNet.2016.9.

[22] A. S. Jose, L. R. Nair, and V. Paul, "Towards detecting flooding DDOS attacks over software defined networks using machine learning techniques," *Revista Gestão Inovação e Tecnologias*, vol. 11, no. 4, pp. 3837–3865, 2021, doi: 10.47059/revistageintec.v11i4.2411.
[23] W. Foundation, "Wireshark Â· Go deep.," Wireshark Foundation, 2016. Online. [Available]: https://www.wireshark.org/.
[24] H. Noman and M. Jasim, "POX Controller and Open Flow Performance Evaluation in Software Defined Networks (SDN) Using Mininet Emulator," in *IOP conference series: materials science and engineering*, 2020, vol. 881, no. 1, p. 012102, doi: 10.1088/1757-899X/881/1/012102.
[25] POX, "Installing POX-POX Manual Current documentation," 2018. [Online]. Available: https://noxrepo.github.io/pox-doc/html/. (Accessed Jun. 07, 2022).
[26] S. Oshima, T. Nakashima, and T. Sueyoshi, "Early DoS/DDoS detection method using short-term statistics," in *2010 International Conference on Complex, Intelligent and Software Intensive Systems*, 2010, pp. 168–173.

BIOGRAPHIES OF AUTHORS

Mohammed Ibrahim Kareem his Bachelor's degree in Information Technology from the University of Babylon, Iraq, in 2013, and his Master's degree in Data Mining from the University of Babylon, Iraq, in 2019. He currently holds a Ph.D. Information security student at Babylon University, Iraq. Worked for several years as a software engineer at OMNNEA Telecom from 2014 to 2020. His research interests include computer networking, network security, and data mining. He can be contacted at email: mohamed.ibrahim@uobabylon.edu.iq.

Mahdi Nsaif Jasim is Assit. Prof. Dr. Mahdi Nsaif Jasim, University of Information Technology and Communications, College of Business Informatics Dept. of Management Information Systems. Born in Babylon, Iraq, lives in Baghdad. Interest: information systems, data and information security, mining in vector data, GIS, database systems. The researcher has interrest in SDN data acquisition and data processing. [He also Supervised a number of Ph.D. and M.Sc. Students in different Iraqi universities. Dr. Mahdi has been supervised 10 M.Sc. students and 5 Ph.D. students. He taught number os B.Sc. and M.Sc. courses a number of Iraqi universities. He can be contacted at email: mahdimnsaif@uoitc.edu.iq.