

Performance evaluation of RYU controller under distributed denial of service attacks

Mohammed Ibrahim Kareem¹, Mahdi Nsaif Jasim², Hussein Ibrahim Hussein³, Karrar Ibrahim⁴

¹Department of Information Security, University of Babylon, Babylon, Iraq

²Department of Business Informatics, University of Information Technology and Communications, Baghdad, Iraq

³Al-Department of Computer Techniques Engineering, AlSafwa University College, Karbala, Iraq

⁴Department of English, College of Education, Al-Zahraa University for Women, Karbala, Iraq

Article Info

Article history:

Received Jul 6, 2022

Revised Jun 27, 2023

Accepted Jul 2, 2023

Keywords:

CPU consumption

Distributed denial of service

Link latency

RAM exhaustion

RYU controller

Software-defined networking

ABSTRACT

Distributed denial of service (DDoS) attacks have been identified as one of the greatest threats to software-defined networking (SDN) because they are highly effective, hard to detect, and easy to use, and they take advantage of vulnerabilities in which the new architecture still exists. This paper describes one technique for denying the RYU controller's services, which can cause the controller's resources to be depleted if a significant number of packets from various zombie hosts are sent to the controller using spoofed source internet protocol (IP) addresses. In order to demonstrate the impact of the attack, we measure various metrics related to RYU controllers such as its central processing unit (CPU) usage and network throughput. In this work, Mininet was used to simulate the data plane and measure metrics such as random access memory (RAM) usage, CPU load, and link latency.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Mohammed Ibrahim Kareem

Department of Information Security, University of Babylon

Babylon, Iraq

Email: mohamed.ibrahim@uobabylon.edu.iq

1. INTRODUCTION

Software-defined networking has drawn a lot of interest as a way to get around the drawbacks of conventional distributed networks. It has a number of benefits, but the separation of the control plane and data plane-which makes the network infrastructure more flexible and controllable-is its main advantage [1], [2]. Although software-defined networking (SDN) represents a promising future network architecture, distributed denial of service (DDoS) attacks can exploit its centralized configuration settings. Computer networks are significantly at risk from DDoS attacks because they are difficult to detect due to their distributed nature and relatively easy to carry out [3]. Because of this flaw, incoming packets from switches will flood the SDN controller, causing it to overload [4]. The three layers that make up an SDN are the data plane (data plane), the control plane (control plane), and the application plane (apps plane) (management plane) as shown in Figure 1. The control plane is a controller that oversees the network as a whole. Decisions about the management, quality of service (QoS), and security of the network are all made by the controller [5]–[7].

Using open, standardized interfaces known as south-bound interfaces and protocols like OpenFlow, the SDN controller connects with switches (forwarding only devices) in the data plane. To carry out a variety of tasks, including routing, switching, firewalling, network address translation, and load balancing, the controller directs the forwarding elements (data plane switches) [8]. SDN offers an innovative approach to network management. In SDN, switches do not process incoming packets; instead, they merely search their forwarding tables for a match. If no match is found, the packets are forwarded to the controller for processing.

SDN's controller is its operating system. It processes the packets and determines whether they will be forwarded or discarded by the switch. Using this method, SDN divides the forwarding and processing planes [7].

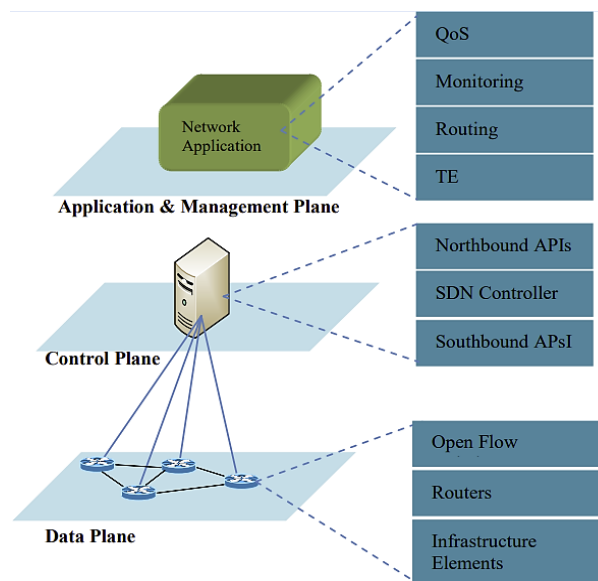


Figure 1. SDN architecture [6]

While this new architecture represents significant gains [9], it is not without flaws and security risks [10]. Several problems have been highlighted, including an absence of authentication measures for packet senders [11]. Due to these flaws, a number of methods have been used to target the controller and endpoints' accessibility, integrity, and confidentiality. DoS and DDoS attacks, in particular the disruption of the controller to prevent switches from processing requests, have been modified to suit the SDN architecture.

A significant risk is posed by the centralized nature of network control in SDN. The effects of these attacks on the controller, if it is compromised, will be felt throughout the entire network. Compared to traditional networks, where control is distributed, this amplifies the effects of such attacks. Consequently, a controller attack in an SDN environment can have a much bigger impact than a comparable attack on a traditional environment.

DoS and DDoS [12], [13] attacks have reportedly been undertaken against a number of SDN controllers [14], including FloodLight, NOX, and RYU [15]-[17], among others, according to past studies. To the best of our knowledge, no in-depth research has been conducted on DoS and DDoS attacks conducted via POX and RYU controllers. Therefore, the primary objective of this research is to look at the impact of a DDoS attack on the RYU controller.

2. RELATED WORK

Ahrammahi and Bhaya [18] employed a Mininet emulator and an OpenFlow switch connected to a POX controller to conduct an experimental test. The aim was to implement and compare the performance of four load balancing algorithms: random, round robin (RR), weighted round robin (WRR), and least connections (LC). Various parameters were considered to evaluate the effectiveness of the comparison process. These included workload, load balancing level, and time-related values such as max-time, min-time, average-time, and response time. These parameters were analyzed to assess the performance and efficiency of the load balancing algorithms being tested.

Mahdi and Abdullah [19] suggest network infrastructure configured using a Mininet network emulator, and the FlowVisor hypervisor is used to virtualize the infrastructure and enable multiple SDN controllers. Additionally, three scenarios are presented for network slicing implementation and comparison. The findings indicate that as the number of tenants increases and the network is divided into more than two slices, the performance of the network approaches and may match that of the network without the use of FlowVisor, which results in the control of the delay rate and the data transfer rate.

Noman and Jasim [20] uses network monitoring tools like Iperf and distributed internet traffic generator (D-ITG) to measure and record performance indicators including service delay, utilized bandwidth,

received packets and bytes in order to examine the controller's capabilities and assess how well they function in an SDN setting. Bhardwaj and Panda [21] describes an implementation of SDN architecture for network traffic analysis using an open-source RYU SDN controller. The proposed work examines the performance of a custom network topology based on SDN architecture for node-to-node performance characteristics such as bandwidth, throughput, and round-trip time.

Abood and Abdul-Majeed [22] employs the entropy-threshold technique, which is programmed in Python within the controllers, as well as DDoS attacks carried out by internet of things (IoT) sensors. If the entropy value falls below a certain threshold, the controller will block that specific port in the switch and shut it down. The results show that FloodLight outperforms RYU controllers in dealing with DDoS attack cases. In addition, the FloodLight controller's Throughput decreased from 0.1158 KB/s to 0.08916 KB/s, while the RYU controller's throughput decreased from 0.0646 KB/s to 0.03294 KB/s.

3. SDN COMPONENTS

SDN can be identified by its five main attributes: separation of planes, streamlined devices, centralized control, network automation, and virtualization, all of which promote openness [22]. OpenFlow is essential component in SDN architecture. It is used to packet forwarding or flow statistics collection. OpenFlow switches, a secure channel, and a controller [23] make up the bulk of the architecture. Flow tables are maintained by the devices, better known as switches. Flow, counters, and actions can be uniquely identified for each entry using the several match fields that are provided for this purpose. It is possible to define a flow as a group of packets that are all in the same time period and have the same properties. An optional encryption technology known as transport layer security (TLS) can be used to encrypt data between data plane and control plane [24]. The controller is in charge of maintaining the flow tables on the switch. It is possible for the controller to make OpenFlow requests or add/remove entries from the flow tables.

Figure 2 depicts the packet forwarding mechanism in an OpenFlow switch. The header packet are captured and inspected when the packet arrives at the switch for matching reasons. To find a match, the flow rules recorded in the flow table are compared to the header packet in ascending order using the flow rules. To begin, the matching system looks for a flow rule that matches this packet to an existing defined flow; if it does, the switch performs the action described in this rule. If a match is not found, the switch delivers the packet information to the controller in an OpenFlow packet called packet in, which is then processed. Once this flow has been identified, the controller creates a new switch rule for this flow [23].

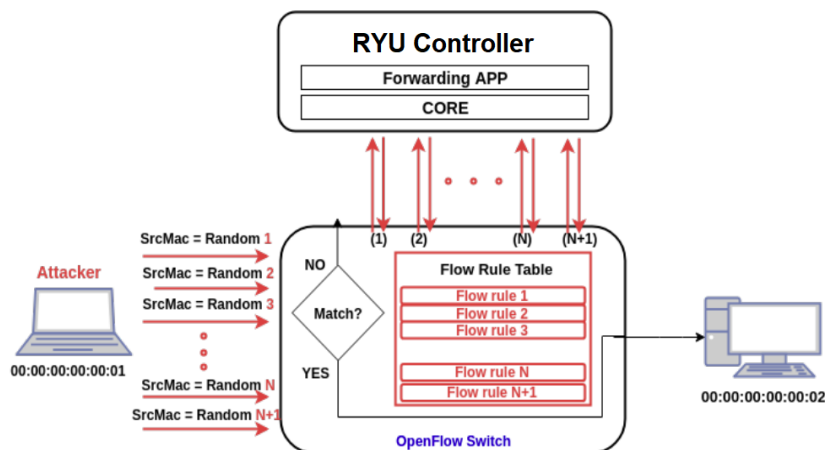


Figure 2. Packets forwarding in OpenFlow switch

4. LAUNCH DDoS ATTACK IN SDN

The RYU controller is the target of a DDoS attack, which is DoS initiated by many compromised hosts (zombies) known as zombies [25]. To carry out a DoS attack, the attacker takes advantage of the controller's limited resources. The DDoS strategy entails flooding the controller with a large number of fabricated packets in order to deplete its available resources, such as central processing unit (CPU) processing power and network bandwidth as shown in Figure 2. During our testing, we discovered that as each packet is processed, the RYU forwarding function generates a new flow rule with the source and destination MAC

addresses as matching criteria. The DoS attack is based on a heuristic that focuses on overwhelming the controller by sending a large number of fabricated packets in a continuous stream. This places an undue strain on the controller's resources, as each packet causes the RYU forwarding application to create a new flow rule. This accumulation of flow rules and associated processing tasks can eventually exhaust the controller's CPU and network bandwidth, resulting in severe performance degradation and a negative impact on overall network operation.

It is common knowledge that a single Python script was written utilizing the Scapy package. It's an advanced user interface for manipulating packets. It can send, receive, alter, and decode packets of many different protocols, as well as match requests with their responses. Scanning, tracing, probing, unit testing, assaults, and network discovery are just some of the commonplace activities it can accomplish with ease (it can replace hping, 85% of nmap, arpspoof, arp-sk, arping, tcpdump, tshark, and p0f,) [26] as shown in Figure 3.

When the fabricated packets reach any switch in this attack scenario, they must be forwarded to the controller as a single packet. The attacker sends packets from multiple compromised hosts, with the goal of exceeding the controller's threshold by flooding it with a high influx of fabricated packets every second. In this attack, internet control message protocol (ICMP) packets are preferred over user datagram protocol (UDP) packets. The reason for this selection is that the Scapy tool used for packet creation and transmission spends less time generating and transmitting ICMP packets, allowing for a higher transmission rate.

```

"Node: h1"@mininet-vm
<Ether type=0x800 |<IP frag=0 proto=udp src=29.120.184.5 dst=['10.0.0.8'] |<UD
P sport=http dport=1 |>>>
.
Sent 1 packets.
139.54.218.168
<Ether type=0x800 |<IP frag=0 proto=udp src=139.54.218.168 dst=['10.0.0.8'] |<UDP sport=http dport=1 |>>>
.
Sent 1 packets.
32.44.180.176
<Ether type=0x800 |<IP frag=0 proto=udp src=32.44.180.176 dst=['10.0.0.8'] |<UDP sport=http dport=1 |>>>
.
Sent 1 packets.
25.189.160.164
<Ether type=0x800 |<IP frag=0 proto=udp src=25.189.160.164 dst=['10.0.0.8'] |<UDP sport=http dport=1 |>>>
.
Sent 1 packets.
14.158.123.252
<Ether type=0x800 |<IP frag=0 proto=udp src=14.158.123.252 dst=['10.0.0.8'] |<UDP sport=http dport=1 |>>>
.
Sent 1 packets.
144.77.246.199
<Ether type=0x800 |<IP frag=0 proto=udp src=144.77.246.199 dst=['10.0.0.8'] |<UDP sport=http dport=1 |>>>
.
Sent 1 packets.
62.225.40.89
<Ether type=0x800 |<IP frag=0 proto=udp src=62.225.40.89 dst=['10.0.0.8'] |<UDP sport=http dport=1 |>>>
.
Sent 1 packets.
210.106.82.137
<Ether type=0x800 |<IP frag=0 proto=udp src=210.106.82.137 dst=['10.0.0.8'] |<UDP sport=http dport=1 |>>>
[]

```

Figure 3. Traffic generation using Scapy

5. EVALUATION

It will be clarified as SDN environment and the remaining components where it is a type of flood attack, such as the Scapy tool, to display the outcomes of the testing phase. Multiple packets are sent to the network device in order to interrupt or slow down the service. The CPU and random-access memory (RAM) consumption levels, as well as the packet delay rate across the attack connection, are then computed. The subsections that follow will detail the tests, measurements, and results.

5.1. Testbed

A laptop with the following specifications was utilized for the tests in an emulated environment: Windows 11, Core i5 11th generation, 16 GB RAM, two virtual machines running Ubuntu-16.04.2-server-amd64, and Mininet 2.2.2. Figure 4 depicts the testbed topology. The RYU controller was installed on a virtual system with 2 core CPU and four gigabytes of RAM. Mininet's topology consists of 5 switches and 8 hosts. The Mininet was housed in a virtual computer with two core and four gigabytes of RAM.

A maximum pace of 100 packets per second were allowed in each attacking host's setup, or 4.2 kilobytes per second per attacking host. Despite the Scapy library's delayed creation and transmission of spoof packets, it is sufficient in this testbed to observe the effects of a DDoS attack. The measurements were taken

over a 4-minute period, with the attack launched by a single host using a spoofed internet protocol (IP) address. It's worth noting that the DDoS launched at the very first second of the test. Table 1 summarizes the details to provide an overview of the average rate of packets arriving at the controller for processing in each attack scenario. Figure 4(a) presents the CPU usage of the RYU controller in two scenarios: before the DDoS attack and during the attack, while Figure 4(b) focuses on the network utilization of the RYU controller before the DDoS attack and during the attack. Figure 5 presents a comparison of CPU and RAM usages in the Mininet (data plane side).

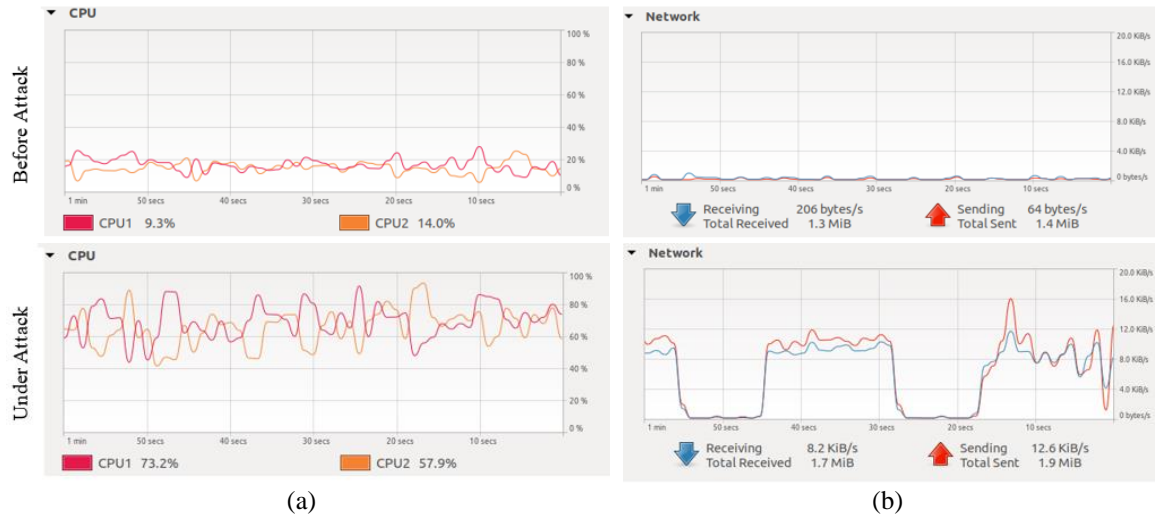


Figure 4. Before and under attack for control plane of RYU controller (a) RYU CPU usage and (b) RYU network utilize

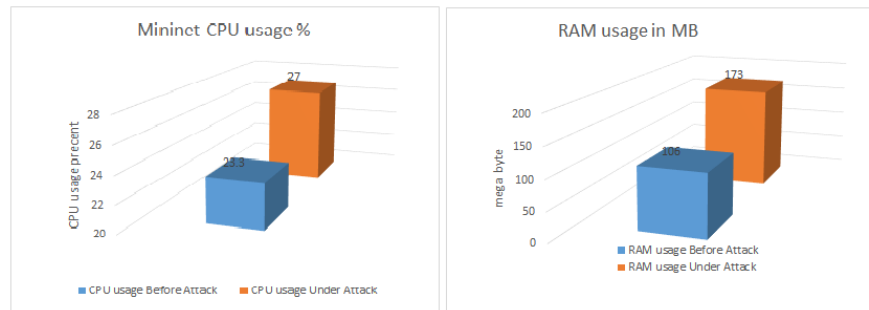


Figure 5. CPU and RAM of mininet before and under attack

Table 1. Average link latency between h1 and h2

Before attack	Under attack
0.5334 ms	4.7465

5.2. Metrics

Both the CPU utilization and the data transfer rate on the control plane were measured before and after the attack to demonstrate the impact of the attack on a network. On the other side, link delay between data plane hosts was estimated, both before and during DDoS attacks, along with the RAM and CPU consumption. Mininet CPU use, link delay, and RAM usage in the data plane are the measures used to gauge attack performance.

Additionally, the control plane's network bandwidth utilization and CPU usage by the RYU controller. The Linux monitor tool was employed to measure CPU usage, and it takes a snapshot of the percentage of CPU usage at any given time every two seconds and stores it. The duration required for a packet to traverse a specific link is referred to as link latency. In this research paper, link latency was measured between hosts 1 and 2 under

two conditions: before and during the attack as shown in Figure 6. The authors devised a strategy for measuring link latency using the Mininet CLI monitor, as shown in Figure 7.

Due to the flow rule that was previously instantiated at the switch, once the packet had completed its journey across the link, it was returned to the controller. The link latency was calculated by calculating the total time required for the UDP packet to complete a round trip between hosts 1 and 2. Figure 8 explain the SDN topology using the RYU controller, the RYU controller acts as the central management entity that controls the behavior of the network switches.

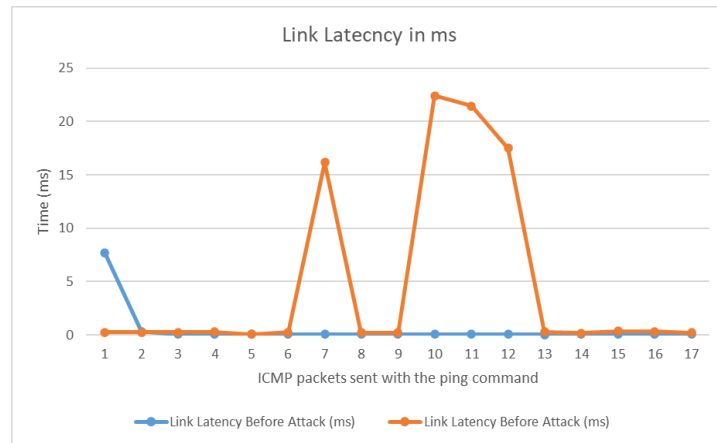


Figure 6. Link latency in ms at data plane (mininet)

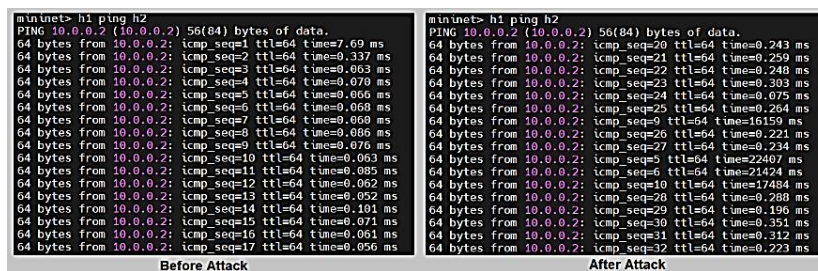


Figure 7. Link latency calculation by PING command between host 1 and host 2

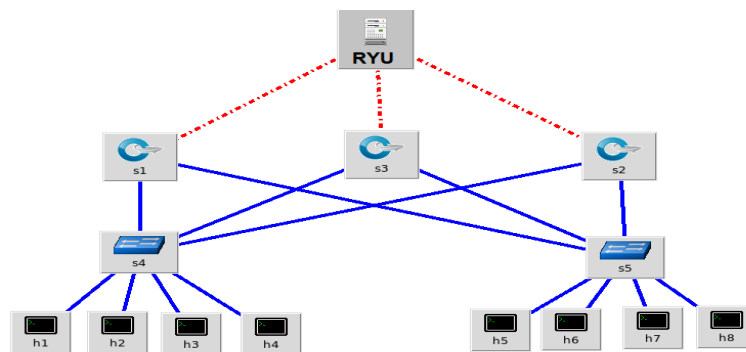


Figure 8. Evaluation of SDN topology using RYU controller

5.3. Results

Observed that the measurement of link latency was conducted by monitoring the Mininet terminal. However, we discovered that the application stopped recording data after 12 seconds of simulation time due to a controller jam, as indicated by the blue and orange vertical lines on the right side of Figure 4. This served as

an indicator of when the controller started to malfunction. Notably, the simulation with four attacking hosts resulted in the shortest time before the application failure, which was approximately 4 minutes.

Examining the CPU usage in each of the three scenarios shown on the left of Figures 4 and 5, we observed that as the number of attacking hosts increased, the fluctuation in CPU consumption decreased. It tended to reach a constant maximum value, effectively saturating the controller. The scenario with the DDoS attack caused the controller's CPU consumption to exceed 60%, representing the most severe case in terms of resource utilization. Furthermore, the time taken for a UDP packet to traverse the link between host 1 and host 2 increased significantly under the attack, as shown in Figures 6 and 7. This signifies a substantial delay in packet circulation within the network, particularly when the attack is actively deployed.

6. CONCLUSIONS AND FUTURE WORK

Finally, this paper concentrated on the use of a flooding attack in an SDN network to disrupt the services of a RYU. The results highlighted the severe impact of the attack on CPU consumption, network performance and RAM usage in the controller and network traffic delay. The findings showed that as the number of attacking hosts increased, so did the controller's workload. When DDoS attack in an SDN launch was determined by the RYU controller's available resources and the attack power in terms of the number of compromised hosts.

In the future, developing an intrusion detection system (IDS) within the controller as a future project. This IDS would classify network traffic and detect anomalies using software-based techniques and machine learning algorithms. Overall, this study contributes to a better understanding of the vulnerabilities and consequences of DDoS attacks in SDN architectures, emphasizing the importance of robust defense mechanisms to protect the controller while maintaining network integrity and performance.





REFERENCES

- [1] M. I. Kareem and M. N. Jasim, "Machine learning-based DDoS attack detection in software-defined networking," in *International Conference on New Trends in Information and Communications Technology Applications*, 2022, doi: 10.1007/978-3-031-35442-7_14.
- [2] K. F. Hassan and M. E. Manaa, "Detecting distributed denial of service attacks in internet of things networks using machine learning in fog computing," in *2022 5th International Conference on Engineering Technology and its Applications (IICETA)*, 2022, pp. 323–328, doi: 10.1109/IICETA54559.2022.9888699.
- [3] 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 (BEEI)*, vol. 11, no. 3, 2022, doi: 10.11591/eei.v11i3.3643.
- [4] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceeding IEEE*, vol. 103, no. 1, pp. 14–76, 2014, doi: 10.1109/JPROC.2014.2371999.
- [5] S. Askar, "SDN-based load balancing scheme for fat-tree data center networks," *Al-Nahrain Journal for Engineering Sciences*, vol. 20, no. 5, pp. 1047–1056, 2017.
- [6] S. K. Askar, "Adaptive load balancing scheme for data center networks using software defined network," *Science Journal of University of Zakho*, vol. 4, no. 2, pp. 275–286, 2016, doi: 10.25271/2016.4.2.118.
- [7] 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)*, 2021, pp. 29–34, doi: 10.1109/IT-ELA52201.2021.9773744.
- [8] G. A. Qadir and S. Askar, "Software defined network based VANET," *International Journal of Business and Social Science*, vol. 5, no. 3, pp. 83–91, 2021.
- [9] H. M. Mohammad and A. A. Abdullah, "DDoS attack mitigation using entropy in SDN-IoT environment," in *AIP Conference Proceedings*, 2023, vol. 2591, no. 1, doi: 10.1063/5.0123465.
- [10] A. F. Ali and W. S. Bhaya, "Software defined network (SDN) security against address resolution protocol poisoning attack," *Journal of Computational and Theoretical Nanoscience*, vol. 16, no. 3, pp. 956–963, 2019, doi: 10.1166/jctn.2019.7982.
- [11] C. Yoon *et al.*, "Flow wars: systemizing the attack surface and defenses in software-defined networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3514–3530, 2017, doi: 10.1109/TNET.2017.2748159.
- [12] M. I. Kareem and M. N. Jasim, "Fast and accurate classifying model for denial-of-service attacks by using machine learning," *Bulletin of Electrical Engineering and Informatics (BEEI)*, vol. 11, no. 3, pp. 1742–1751, 2022, doi: 10.11591/eei.v11i3.3688.
- [13] M. I. Kareem and M. N. Jasim, "DDoS attack detection using lightweight partial decision tree algorithm," in *2022 International Conference on Computer Science and Software Engineering (CSASE)*, 2022, pp. 362–367, doi: 10.1109/CSASE51777.2022.9759824.
- [14] R. T. Kokila, S. T. Selvi, and K. Govindarajan, "DDoS detection and analysis in SDN-based environment using support vector machine classifier," in *2014 Sixth International Conference on Advanced Computing (ICoAC)*, 2014, pp. 205–210, doi: 10.1109/ICoAC.2014.7229711.
- [15] A. N. Kahdim and M. E. Manaa, "Design an efficient internet of things data compression for healthcare applications," *Bulletin of Electrical Engineering and Informatics (BEEI)*, vol. 11, no. 3, pp. 1678–1686, 2022, doi: 10.11591/eei.v11i3.3758.
- [16] M. E. Manaa and M. S. Abedoun, "Securing of software-defined networking (SDN) from DDoS attack using a blockchain," in *Next Generation of Internet of Things: Proceedings of ICNGIoT 2021*, 2021, pp. 397–411, doi: 10.1007/978-981-16-0666-332.
- [17] A. J. Altamemi, A. Abdulhassan, and N. T. Obeis, "DDoS attack detection in software defined networking controller using machine learning techniques," *Bulletin of Electrical Engineering and Informatics (BEEI)*, vol. 11, no. 5, pp. 2836–2844, 2022, doi: 10.11591/eei.v11i5.4155.
- [18] M. A. M. Alrammahi and W. S. Bhaya, "Performance analysis for load balancing algorithms using POX controller in SDN," in *2022 International Conference on Data Science and Intelligent Computing (ICDSIC)*, 2022, pp. 175–180, doi: 10.1109/ICDSIC56987.2022.10076081.
- [19] S. S. Mahdi and A. A. Abdullah, "Implementation of network slicing for multi-controller environment based on FlowVisor," in *International Conference on New Trends in Information and Communications Technology Applications*, 2022, pp. 173–188, doi: 10.1007/978-3-031-35442-7_10.





- [20] H. M. Noman and M. N. Jasim, "POX controller and open flow performance evaluation in software defined networks (SDN) using mininet emulator," in *IOP Conference Series: Materials Science and Engineering*, vol. 881, no. 1, 2020, p. 12102, doi: 10.1088/1757-899X/881/1/012102.
- [21] S. Bhardwaj and S. N. Panda, "Performance evaluation using RYU SDN controller in software-defined networking environment," *Wireless Personal Communications*, vol. 122, no. 1, pp. 701–723, 2022, doi: 10.1007/s11277-021-08920-3.
- [22] M. J. K. Abood and G. H. Abdul-Majeed, "Classification of network slicing threats based on slicing enablers: A survey," *International Journal of Intelligent Networks*, 2023, doi: 10.1016/j.ijin.2023.04.002.
- [23] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using openflow: A survey," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 1, pp. 493–512, 2013, doi: 10.1109/SURV.2013.081313.00105.
- [24] S. S. Mahdi and A. A. Abdullah, "Improved security of SDN based on hybrid quantum key distribution protocol," in *2022 International Conference on Computer Science and Software Engineering (CSASE)*, 2022, pp. 36–40, doi: 10.1109/CSASE51777.2022.9759635.
- [25] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004, doi: 10.1145/997150.997156
- [26] "Scapy." 2023, scapy.net. <https://scapy.net/> (accessed Jun. 07, 2022).

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 technology at Babylon University, Iraq. Worked for several years as a software engineer at OMNNEA Telecom from 2014 to 2020. His research interests include SDN, 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 interest in SDN data acquisition and data processing. He also Supervised a number of Ph.D. and MSc. Students in different Iraqi universities. Dr. Mahdi has been supervised 10 MSc students and 5 Ph.D. students. He taught number of B.Sc. and M.Sc. courses a number of Iraqi Universities. He can be contacted at email: mahdimnsaif@uoitc.edu.iq.



Hussein Ibrahim Hussein     was born on November 11 in Iraq, received the B.S. degree in computer engineering, in 2014 and the M.S. degree in Embedded System Design Engineering from University Malaysia perils in 2017 and the Doctor of Engineering degree from Perlis University, Malaysia in 2021. His current research interests include embedded system and artificial intelligence. He can be contacted at email: hussein.sarhan@alsafwa.edu.iq.



Karrar Ibrahim     a Bachelor's degree in computer technology engineering from Al-Mansour University College, Iraq, in 2015, and his Master's degree in Network Technology from UTeM, Malaysia, in 2018. He is currently a Ph.D. in Information technology student at Babylon University, Iraq. Worked for several years as an IT engineer at Al Liwa International Company from 2015 to 2017 and also at Al-Zahraa university for women as an IT manager from 2020 to present. His research interests include computer networking, communication, and machine learning. He can be contacted at email: karrar.ibrahim@alzahraa.edu.iq.