

# Simulation of SDN in mininet and detection of DDoS attack using machine learning

P. Karthika, Karmel Arockiasamy

School of Computing Science and Engineering, Vellore Institute of Technology, Chennai, India

## Article Info

### Article history:

Received Dec 15, 2022

Revised Dec 25, 2022

Accepted Jan 10, 2023

### Keywords:

Detection

Machine learning

Mininet

Ryu

Software defined networking

Distributed denial of service

## ABSTRACT

Most contemporary businesses are embracing software defined networking (SDN), a developing architecture that enables an aerial-like perspective of the entire network. SDN operates by virtualizing the network and provides advantages including improved performance, visibility, speed, and scalability. SDN attempts to divide the network control plane from the forwarding plane. The control plane, which includes one or more controllers and incorporates complete intelligence, is thought of as the brain of the SDN. However, SDN has challenges with controller vulnerability, flexibility, and hardware security. But distributed denial of service (DDoS) assaults constitutes a serious threat to the SDN. Transmission control protocol-synchronized (TCP-SYN) floods, a common cyberattack that can harm SDNs, can deplete network resources by opening an excessive number of illegitimate TCP connections. In this research, we provide an OpenFlow port statistic-based architecture for machine learning (ML) enabled TCP-SYN flood detection. This research showed that ML models like support vector machine (SVM), Navie Bayes, and multi-layered perceptron can distinguish between regular traffic and SYN flood traffic and can mitigate the impacts of the attacking node on the network. Results showed that the multilayered perceptron can classify the traffic with highest accuracy of 99.75% for the simulation dataset.

This is an open access article under the [CC BY-SA](#) license.



## Corresponding Author:

Karmel Arockiasamy

School of Computer Science and Engineering, Vellore Institute of Technology

Chennai, Tamil Nadu 600127, India

Email: karmel.a@vit.ac.in

## 1. INTRODUCTION

The software defined networking (SDN) evolving as the hopeful architecture for the future network and it has the benefits of global view and central control programmability [1], [2]. Recently, the SDN has shown much attention and capable of producing an efficient distributed denial of service (DDoS) defense against different forms of DDoS attack based on network [3], [4]. The centralized controller could influence the knowledge of its own network in the SDN architecture to detect DDoS attacks through various techniques such as machine learning (ML) or analysis of traffic patterns. Whenever detected the DDoS attack, the controller SDN tries to block the attackers flow or redirect the original traffic to a protected system by sending out and using an updated security policy for network switches [5]–[7].

DDoS attacks have drawn increased attention to the internet during the last five years. In order to safeguard the network from threats, intrusion detection systems (IDS) are widely used in large networks [8]. Recently, the techniques and concepts of SDN were implemented and researched [9], [10]. The central controller was constructed based on the software logical set to manage data plane via southbound application programming interface (API) [11] and provides the network services to the management plane via

northbound API. Furthermore, the management plane includes different business applications like traffic engineering applications and virtual network security applications [12].

The architectural difference between the traditional network and the SDN network would threaten the availability of the SDN by DDoS attacks [13]. Particularly, the DDoS attacks affect the SDN controller. The goal of a denial of service (DoS) attack is often to deplete the network's resources before blocking access to it for targeted users [14], [15]. The DoS attack on SDN uses different logic in control plane, data plane and determine a network scanning tool for identifying the SDN network. SDN has programmability and centralized control; still, the network is facing different security challenges and among them, the DDoS attack causes a serious threat [16]. Mostly, DDoS attack would damage the certain resources of services or intend user connections like server resources and network resources [17]. Different ML was playing a major role in detecting the presence of attack in SDN, particularly, the neural network (NN), support vector machine (SVM) and so on. In this study, the DDoS attacks are categorized using ML models. The work's contributions are as follows:

- The SDN-specific dataset for both regular and attack flows is created using the mininet emulator.
- Recording numerous differential port statistics under both normal and attack conditions to simulate both situations, while eliminating class imbalance by assuring equitable representation.
- ML classifiers training on our data to generalize both situations.
- Valuating the effectiveness of our method for detecting transmission control protocol-synchronized (TCP-SYN) floods in real-time.

## 2. SOFTWARE DEFINED NETWORK

The SDN has recently emerged as the next big thing in the networking business. The separation of the control plane from the data plane is what makes the SDN special. Figure 1 represents the data plane, control plane, and application plane components of the SDN architecture. The first layer is the infrastructure layer composed of network devices like OpenFlow switches and routers. These network devices have the control to forward packets in the network [18]. The main function of network devices is to forward the user packets to the next switch with the routing information gleaned from the data plane packets.

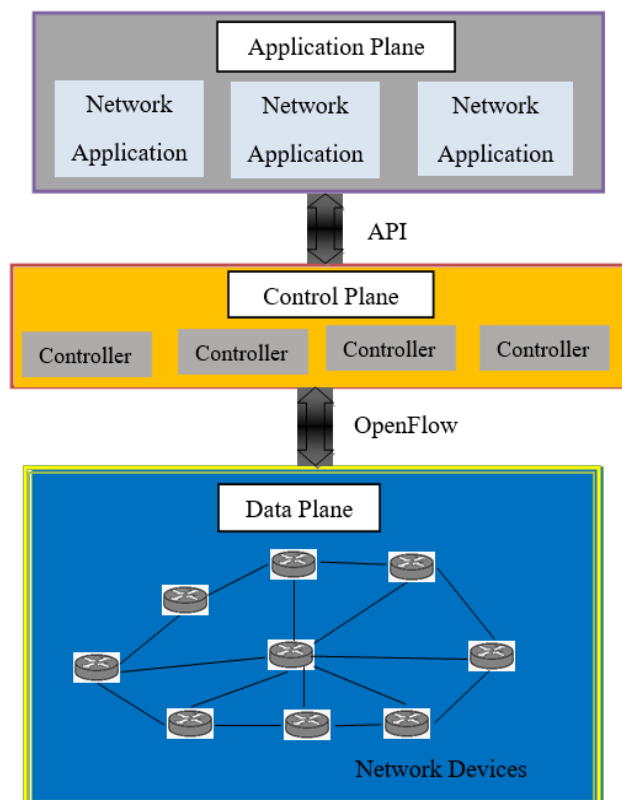


Figure 1. SDN architecture

The second layer is the network operating system, which is to configure the network devices. The control plane contains one or more controllers like open network operating system (ONOS), Maestro, Ryu and OpenDayLight, OpenFloodLight. These controllers are considered as the brain of the SDN networks. The forwarding decision for new flow can decide and configured by the central controller [18], [19]. SDN connected with the OpenFlow protocol, is the first standard for the southbound protocol. Initial specifications were published by Stanford's clean slate initiative; the open network foundation (ONF) now maintains them. The third layer of SDN is the application layer, which contains application program that communicate with the SDN controller to support the operation of forwarding process [20]. An application can aggregate information from the controller to create an abstract network view for decision-making.

### 3. DISTRIBUTED DENIAL OF SERVICE ATTACKS

The impact of successful DDoS attacks is a result of the growing need for the internet. Multiple workstations launch DDoS attacks against a single server by flooding its IP address with a lot of traffic [20]. Using the internet principle, this attack attempts to disrupt the targeted system thus overwhelming legitimate users by denying them access to the system resources. In recent years, detection of DDoS attacks become more difficult, where attackers use multiple protocol for the DDoS attacks are common [21]. Table 1 provides a list of common DDoS attack types.

Table 1. Some of the common DDoS attack types

Attack type	Description
TCP SYN Flood [22]	TCP SYN flood spoof the source IP and quickly overload the target server with low traffic.
Internet control message protocol (ICMP) flood [23]	In ICMP flood, the attacker used ICMP echo request packets to target the victim in order to severely overwhelm their resources and slow the targeted network.
User datagram protocol (UDP) flood [24]	One of the most common network floods is the UDP flood, any DDoS attack that flood single destination or random target with the user datagram protocol packets. In most cases the attackers spoof the source IP, which is easy to do since the UDP is the "connectionless" and does not follow the handshaking procedure. The main intention of UDP is to saturate the internet pipe.
Zero-day flood [25]	The "zero-day" used to describe all unknown attacks or new attacks exploiting vulnerabilities.

## 4. SIMULATION OF SDN NETWORK USING THE RYU CONTROLLER

### 4.1. Topology creation

In this section, a network simulation study using the Ryu controller and SDN in a virtualized environment with mininet will be shown, and the outcomes will be assessed. Using the Python script "topology1.py," a topology with four hosts, three switches, and one controller are generated for the simulation scenario as depicted in Figure 2. The next step is to launch the Ryu controller as shown in Figure 3, so that it can connect to the virtual network SDN from their own mininet virtual machine or from a host machine once the topology has been established.

The next step is to test the connectivity as shown in Figure 4 by executing the simple ping command. The fundamental TCP/IP command, "ping," is used to check and verify that a given destination IP address is real and capable of responding to requests for computer network administration. The switching hub is currently in the position of waiting for packet-in after OVS was connected, the handshake was completed, the table-miss flow entry was added as displayed in Figure 5. The transfer data size is specified as 65535 (0xffff=OFPCML NO BUFFER) and the priority level is specified as 0, no match.

```

Karthika:~/Desktop/src$ sudo python3 topology1.py
[sudo] password for test:
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 server1 server2
*** Adding switches:
s1 s2 s3
*** Adding links:
(10.00Mbit) (10.00Mbit) (h1, s2) (10.00Mbit) (10.00Mbit) (h2, s2) (10.00Mbit) (10.00Mbit) (h3, s2) (10.00Mbit) (10.00Mbit) (h4, s2) (10.00Mbit) (10.00Mbit) (s1, s2) (10.00Mbit) (10.00Mbit) (s1, s3) (10.00Mbit) (10.00Mbit) (server1, s3) (10.00Mbit) (10.00Mbit) (server2, s3)
*** Configuring hosts
h1 h2 h3 h4 server1 server2
*** Starting controller
c1
*** Starting 3 switches
s1 s2 s3 ... (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit)
*** Starting CLI:

```

Figure 2. Topology creation

```
Karthika:~/Desktop/src$ ryu-manager app.py
loading app app.py
loading app ryu.controller.ofp_handler
instantiating app app.py of DDOSMLApp
instantiating app ryu.controller.ofp_handler of OFPHandler
start flow monitoring thread
flow collection interval 30
monitor the flows...
monitor the flows...
monitor the flows...
monitor the flows...
monitor the flows...
monitor the flows...
monitor the flows...
```

Figure 3. Ryu controller

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data:
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=6.56 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.281 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.049 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.055 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.075 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.056 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.064 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.089 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.055 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.075 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.175 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.054 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.061 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=0.086 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=0.063 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=0.057 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=0.059 ms
```

Figure 4. Ping command

```
Karthika:~/Desktop/src$ sudo ovs-ofctl -O OpenFlow13 dump-flows s1
[sudo] password for test:
cookie=0x0, duration=1.571s, table=0, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
```

Figure 5. Table miss flow entry

## 4.2. Traffic flow

### 4.2.1. Normal traffic flow

The ICMP, TCP, and UDP protocols are used to create the normal traffic flow, as depicted in Figures 6-8. A well-known tool for testing networks is Iperf, which can create TCP and UDP data streams and measure the network's throughput while transmitting them. The UDP traffic is created using the command below. For 300 seconds, UDP traffic was generated. Iperf-u-c server1-t 300.

```
"Node: h1"
root@karthika:/home/test/Desktop/src# iperf -u -c 10.0.0.5 -t 300
-----
Client connecting to 10.0.0.5, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size: 208 KByte (default)
-----
[ 5] local 10.0.0.1 port 51605 connected with 10.0.0.5 port 5001
ID] Interval      Transfer      Bandwidth
[ 5] 0.0-300.0 sec  37.5 MBytes  1.05 Mbits/sec
[ 5] Sent 26750 datagrams
[ 5] Server Report:
[ 5] 0.0-300.0 sec  37.5 MBytes  1.05 Mbits/sec  2.118 ms  0/26750 (0%)
root@karthika:/home/test/Desktop/src#
```

Figure 6. UDP traffic

The TCP client (-c) started at h2 with port 80 (-p). After -c, server's internet protocol address need to be specified. The following command is used to generate TCP traffic. The TCP traffic was created for 300 seconds. Iperf-c server1-t 300-p 80.

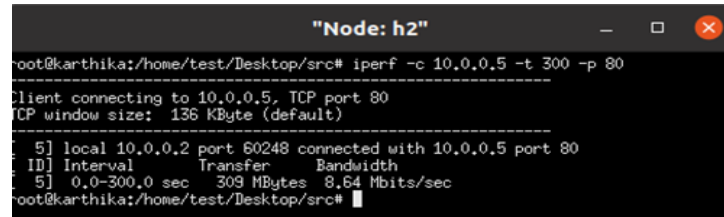


Figure 7. TCP traffic

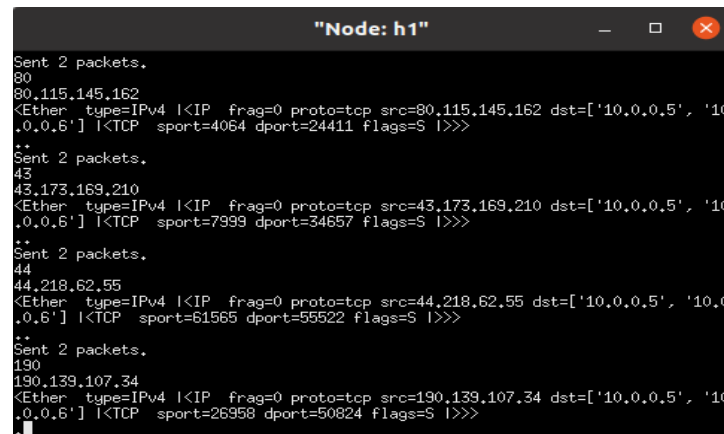


Figure 8. SYN attack flow

#### 4.2.2. Attack flow

The attack traffic flow is created using a TCP SYN flood. To consume resources on the targeted server, it takes advantage of the common TCP three-way handshake. Network saturation results from the attacker sending TCP connection requests more quickly than the server can handle them. The attack is launched using following code: `python3-E synattack.py server1 server2`. The flow statistics are collected for every 30 seconds as showed in Figure 9. The SDN controller sends OpenFlow stats request message to all switches to gather its most recent set of statistics for flows and ports. The switches will reply with flow statistics.

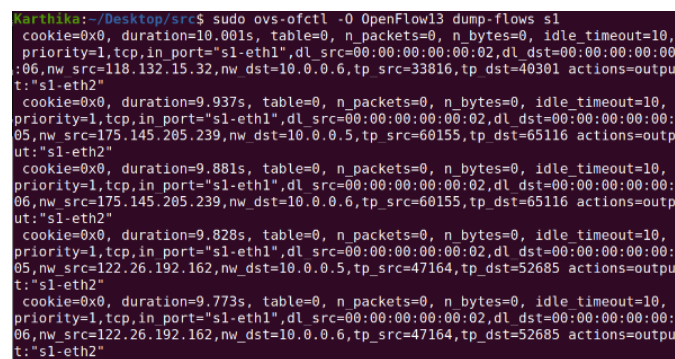


Figure 9. Flow statistics

#### 4.3. Data gathering

The steps to gather the OpenFlow switch flow entries are shown in Algorithm 1. Data on network traffic from the targeted server is gathered for both regular flow and assault flow. The "result.csv" file is updated with those flow details. The target value has been added and given the label "type" to distinguish between the normal flow and the attack flow; a value of 0 has been added for the normal flow and a value of 1 for the attack flow.

**Algorithm 1: Data Gathering**

Begin

One of the target servers should receive traffic from two or more hosts.

observes the passage of s1, s2, and s3

Save the flow to a result.csv file.

Add all feature values to the result.csv file

Add the target value to the file labelled "type," adding 0 for normal flow and 1 for attack traffic.

Read the csv file that has been processed.

Provide the prepared model with the changed csv file.

End

**4.4. Dataset description**

There are total of 11,545 rows and 7 columns in the simulation dataset. The attribute values of the last column named "type" specify the attack type. The value '0' is added for normal traffic and '1' for the attack traffic. This "type" features are used to distinguish the traffic as normal or malicious flow. Table 2 provides a detailed explanation of the features proposed in the dataset.

Table 2. Simulated dataset features

S. No	Features
1	flow_duration
2	ip_proto
3	srcport
4	dstproto
5	packet_count
6	byte_count

**5. DETECTION AND MITIGATION OF DDOS ATTACKS IN SDN USING ML ALGORITHM**

The combination of SDN and a ML algorithm for categorising network traffic is explained in the suggested system model. The categorization model is constructed using supervised learning method. The features of a dataset can be used to train ML classification systems. The ML model is trained throughout this process. The trained model may divide the traffic into benign and malicious groups. Further, real-time traffic classification can be done using the trained model. The Figures 10 and 11 shows the result of detecting of normal and attack flow and blocking the attack flow using MLP.

```
Karthika:~/Desktop/src$ ryu-manager app.py
loading app app.py
loading app ryu.controller.ofp_handler
instantiating app app.py of DDOSMLApp
Using MLPClassifier Algo
start flow monitoring thread
flow collection interval 30
monitor the flows...
monitor the flows...
Extract the flow params 10 1 0 0 980 10
prediction result [0]
The result is 0
```

Figure 10. Detecting normal flow

```
Karthika:~/Desktop/src$ ryu-manager app.py
loading app app.py
loading app ryu.controller.ofp_handler
instantiating app app.py of DDOSMLApp
Using MLPClassifier Algo
start flow monitoring thread
flow collection interval 30
monitor the flows...
monitor the flows...
Extract the flow params 9 6 51298 44273 0 0
prediction result [1]
The result is 1
DDos Detected on from 00:00:00:00:00:02 ...blocking it
```

Figure 11. Detecting and blocking the attack flow

## 6. RESULTS AND DISCUSSION

The simulation setup and outcomes from using ML models are covered in this section. The specified task was carried out using Python. Using the generated simulated dataset, the system performance is evaluated. 25% of the gathered data were utilized for testing, while the remaining 75% were used for training. Accuracy, precision, and F1-score have all been evaluated for the projected model.

### 6.1. Simulation setup

Using mininet, the network topology is created. Four hosts, three switches, and one Ryu controller make up the topology. Mininet environment specifications: for all testing and experiments, a DELL Inc. Inspiron15 5,000 computer with the following specifications was used: 8 GB of RAM, Windows 10 64-bit, a 1.00 GHz Intel Core (TM) i5-10th Gen processor, and VirtualBox Oracle VM version 6.0.18 are all required. The guest operating system mininet emulator version 2.3.1b1 on Linux operating system Ubuntu 14.0432 bits with 4,096 MB of RAM and Ryu controller is installed on this machine and is managed by VirtualBox.

### 6.2. Performance analysis

The inter-arrival period between succeeding packets in the simulated dataset's typical traffic was set to 0.5 seconds. The attack traffic is produced with a 0.05-second inter-arrival interval between every pair of subsequent packets. Figures 12-14 illustrate the performance analysis of different ML models.

#### 6.2.1. Accuracy

The ML models are used to identify the correlations and patterns between variables in a dataset, and the correctness of each model is evaluated using the input, or training, data. It is the proportion of packets in the sample that are correctly categorised and labelled to all packets in the sample as indicated in (1). Both legitimate and harmful classes are correctly predicted by the classifier during the traffic evaluation.

$$\text{Detection Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

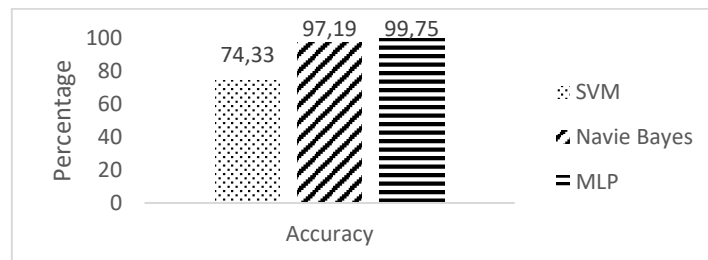


Figure 12. Accuracy

#### 6.2.2. Precision

An accurate prediction made by the model is measured by the precision metric. It is obtained as shown in (2) by dividing the total number of accurate positive forecasts by the total number of true positives. It makes easier to see how trustworthy the ML model is in classifying the model as successful.

$$\text{Precision} = \frac{TP}{TP+FP} \quad (2)$$

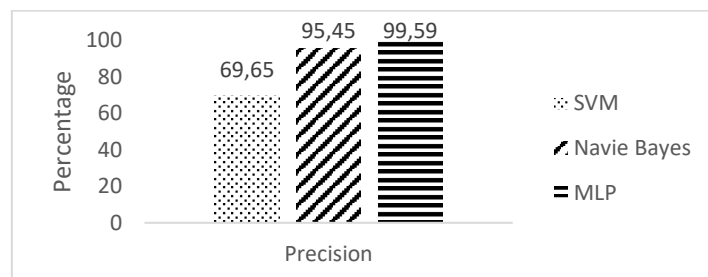


Figure 13. Precision



### 6.2.3. F1-score

The reliability of the test is measured by the F-score or F-measure as indicated in (3). It is based on the test's precision and retrieval. Where precision is the ratio of correctly identified positive responses to all positive results, including those that were not correctly identified, and retrieval is the ratio of correctly identified positive responses to all tests that could have produced positive results.

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (3)$$

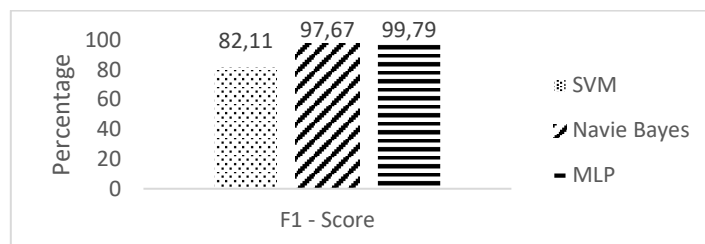


Figure 14. F1-score

### 6.2.4. Confusion matrix

The performance characteristics are calculated using a confusion matrix. It accounts for the two possible forms of errors: false positives, which are benign occurrences that have resulted in a false alarm, and false negatives, which are malevolent instances that have not been detected. It can be understood and interpreted using a 2×2 matrix, where the row represents the predicted labels and the column represents the actual truth labels. Table 3 shows the confusion matrix of the three models.

Table 3. Confusion matrix				
Metrics	MLP	Naïve Bayes	SVM	
True negative	1179	1105	445	
False positive	7	81	741	
False negative	0	0	0	
True positive	1701	1701	1701	

## 7. CONCLUSION

Although the SDN significantly improves networking, it still has several security concerns, with DDoS attacks being the most prevalent one. Due to the SDN controller's role as the system's main control point, the network is particularly susceptible to DDoS attacks. The SDN must therefore have efficient DDoS attack detection. This work provides a comparative analysis of various ML classifiers applied on the simulation dataset for the early and precise detection of DDoS attacks throughout the SDN to address this problem. The benefits of SDN in conjunction with ML classifiers protect the SDN controller from DDoS assaults. The empirical findings demonstrate how well the MLP classifier detects attacks on the SDN controller. In the future, the classifier for detecting DDoS assaults in SDN will be improved using the optimization algorithm.

## REFERENCES




- [1] M. S. El Sayed, N.-A. Le-Khac, M. A. Azer, and A. D. Jurcut, "A flow based anomaly detection approach with feature selection method against DDoS attacks in SDNs," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 4, pp. 1862–1880, Dec. 2022, doi: 10.1109/TCCN.2022.3186331.
- [2] D. Tang, Y. Yan, S. Zhang, J. Chen, and Z. Qin, "Performance and features: Mitigating the low-rate TCP-targeted DoS attack via SDN," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 1, pp. 428–444, Jan. 2022, doi: 10.1109/JSAC.2021.3126053.
- [3] P. Maity, S. Saxena, S. Srivastava, K. S. Sahoo, A. K. Pradhan, and N. Kumar, "An effective probabilistic technique for DDoS detection in OpenFlow controller," *IEEE Systems Journal*, vol. 16, no. 1, pp. 1345–1354, Mar. 2022, doi: 10.1109/JSYST.2021.3110948.
- [4] S. Haider *et al.*, "A deep CNN ensemble framework for efficient DDoS attack detection in software defined networks," *IEEE Access*, vol. 8, pp. 53972–53983, 2020, doi: 10.1109/ACCESS.2020.2976908.
- [5] A. O. Sangodoyin, M. O. Akinsolu, P. Pillai, and V. Grout, "Detection and classification of DDoS flooding attacks on software-






- defined networks: A case study for the application of machine learning," *IEEE Access*, vol. 9, pp. 122495–122508, 2021, doi: 10.1109/ACCESS.2021.3109490.
- [6] K. Kalkan, L. Altay, G. Gur, and F. Alagoz, "JESS: Joint entropy-based DDoS defense scheme in SDN," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2358–2372, Oct. 2018, doi: 10.1109/JSAC.2018.2869997.
- [7] L. Zhu, X. Tang, M. Shen, X. Du, and M. Guizani, "Privacy-preserving DDoS attack detection using cross-domain traffic in software defined networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 628–643, Mar. 2018, doi: 10.1109/JSAC.2018.2815442.
- [8] Y. Yu, L. Guo, Y. Liu, J. Zheng, and Y. Zong, "An efficient SDN-based DDoS attack detection and rapid response platform in vehicular networks," *IEEE Access*, vol. 6, pp. 44570–44579, 2018, doi: 10.1109/ACCESS.2018.2854567.
- [9] K. S. Sahoo *et al.*, "An evolutionary SVM model for DDOS attack detection in software defined networks," *IEEE Access*, vol. 8, pp. 132502–132513, 2020, doi: 10.1109/ACCESS.2020.3009733.
- [10] Y. Wang, T. Hu, G. Tang, J. Xie, and J. Lu, "SGS: Safe-guard scheme for protecting control plane against DDoS attacks in software-defined networking," *IEEE Access*, vol. 7, pp. 34699–34710, 2019, doi: 10.1109/ACCESS.2019.2895092.
- [11] A. B. Dehkordi, M. Soltanaghaei, and F. Z. Boroujeni, "The DDoS attacks detection through machine learning and statistical methods in SDN," *The Journal of Supercomputing*, vol. 77, no. 3, pp. 2383–2415, Mar. 2021, doi: 10.1007/s11227-020-03323-w.
- [12] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against software defined network controllers," *Journal of Network and Systems Management*, vol. 26, no. 3, pp. 573–591, Jul. 2018, doi: 10.1007/s10922-017-9432-1.
- [13] S. Dong, K. Abbas, and R. Jain, "A survey on distributed denial of service (DDoS) attacks in SDN and cloud computing environments," *IEEE Access*, vol. 7, pp. 80813–80828, 2019, doi: 10.1109/ACCESS.2019.2922196.
- [14] M. Conti, C. Lal, R. Mohammadi, and U. Rawat, "Lightweight solutions to counter DDoS attacks in software defined networking," *Wireless Networks*, vol. 25, no. 5, pp. 2751–2768, Jul. 2019, doi: 10.1007/s11276-019-01991-y.
- [15] O. E. Tayfour and M. N. Marsono, "Collaborative detection and mitigation of distributed denial-of-service attacks on software-defined network," *Mobile Networks and Applications*, vol. 25, no. 4, pp. 1338–1347, Aug. 2020, doi: 10.1007/s11036-020-01552-0.
- [16] W. Chen, S. Xiao, L. Liu, X. Jiang, and Z. Tang, "A DDoS attacks traceback scheme for SDN-based smart city," *Computers & Electrical Engineering*, vol. 81, pp. 1–12, Jan. 2020, doi: 10.1016/j.compeleceng.2019.106503.
- [17] K. S. Sahoo, D. Puthal, M. Tiwary, J. J. P. C. Rodrigues, B. Sahoo, and R. Dash, "An early detection of low rate DDoS attack to SDN based data center networks using information distance metrics," *Future Generation Computer Systems*, vol. 89, pp. 685–697, Dec. 2018, doi: 10.1016/j.future.2018.07.017.
- [18] M. M. Oo, S. Kamolphiwong, T. Kamolphiwong, and S. Vasupongayya, "Advanced support vector machine-(ASVM-) based detection for distributed denial of service (DDoS) attack on software defined networking (SDN)," *Journal of Computer Networks and Communications*, pp. 1–12, Mar. 2019, doi: 10.1155/2019/8012568.
- [19] O. Rahman, M. A. G. Quraishi, and C.-H. Lung, "DDoS attacks detection and mitigation in SDN using machine learning," in *2019 IEEE World Congress on Services (SERVICES)*, Jul. 2019, pp. 184–189, doi: 10.1109/SERVICES.2019.00051.
- [20] M. M. Joëlle and Y.-H. Park, "Strategies for detecting and mitigating DDoS attacks in SDN: A survey," *Journal of Intelligent & Fuzzy Systems*, vol. 35, no. 6, pp. 5913–5925, Dec. 2018, doi: 10.3233/JIFS-169833.
- [21] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martinez-del-Rincon, and D. Siracusa, "LUCID: A practical, lightweight deep learning solution for DDoS attack detection," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 876–889, Jun. 2020, doi: 10.1109/TNSM.2020.2971776.
- [22] V. T. Dang, T. T. Huong, N. H. Thanh, P. N. Nam, N. N. Thanh, and A. Marshall, "SDN-based SYN proxy—a solution to enhance performance of attack mitigation under TCP SYN flood," *The Computer Journal*, vol. 62, no. 4, pp. 518–534, Apr. 2019, doi: 10.1093/comjnl/bxy117.
- [23] H. Harshita, "Detection and prevention of ICMP flood DDOS attack," *International Journal of New Technology and Research*, vol. 3, no. 3, pp. 63–69, 2017.
- [24] Z. A. El Houda, L. Khokhi, and A. S. Hafid, "Bringing intelligence to software defined networks: Mitigating DDoS attacks," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2523–2535, Dec. 2020, doi: 10.1109/TNSM.2020.3014870.
- [25] I. Mbona and J. H. P. Eloff, "Detecting zero-day intrusion attacks using semi-supervised machine learning approaches," *IEEE Access*, vol. 10, pp. 69822–69838, 2022, doi: 10.1109/ACCESS.2022.3187116.

## BIOGRAPHIES OF AUTHORS



**P. Karthika**    obtained her B.E degree in Computer Science and Engineering from Anna University, Chennai in 2010, and her M.E degree in Computer Science and Engineering from Anna University of Technology, Coimbatore, in 2012. She is currently pursuing the Ph.D. degree in the School of Computer Science and Engineering at Vellore Institute of Technology, Chennai. Her research interests include network security, software-defined networking, and deep learning. She can be contacted at email: karthika.p2020@vitstudent.ac.in.



**Dr. Karmel Arockiasamy**    is an Associate professor Grade-II, School of Computing Science and Engineering, VIT University, Chennai. She completed her Ph.D in Anna University, 2016. Her area of specialization includes, mobile ad hoc networks, network security, IoT and cyber security. She had published various papers in Scopus indexed journals and guided a greater number of projects. She is an active ACM member. She is a star cyber secure user-R11; she received best faculty award in the year 2011-2012. She is a reviewer in many journals to name a few wireless network, mobile network, and applications. She can be contacted at email: Karmel.a@vit.ac.in.