KOMPUTASI: JURNAL ILMIAH ILMU KOMPUTER DAN MATEMATIKA
Vol. 19 (2) (2022), 78–87 p-ISSN: 1693-7554, e-ISSN:2654-3990

Detection And Classification Of DDos Attack On Software Defined Network

Irma Anggraeni[1*], Dinar Munggaran Akhmad[2]

[1,2] Department of Computer Science, Faculty of Mathematics and Natural Science, Pakuan University, Bogor, West Java, 16143, Indonesia

Abstract

Software-Defined Networking (SDN) is a new network paradigm that changes network architecture. However, it turns out that the SDN network also has several issues, one of which is security. The higher the traffic on the network, the higher the possibility of security threats that will occur. Therefore, it is necessary to detect attacks that might occur on this SDN network. This study will detect attacks on the SDN network with the stages carried out, namely the process of building an SDN architecture using a mininet emulator, then network simulation according to the topology, retrieval of traffic data using wireshark and performing data analysis using the Weka application on the NSL KDD dataset. The results of this research found that a DDos attack a ping of death is an attack that sends messages continuously to the recipient, causing the computer to crash, then analysis of attack classification data is carried out using a dataset to compare machine learning algorithms that have high accuracy was Random Forests. The targeted output in this research is published in the Journal of Computing.

Keywords: *Network attacks; NSL KDD; Software Defined Network (SDN)*

1. Introduction

Software-Defined Networking (SDN) is a new network paradigm that changes network architecture. The existence of an SDN Network allows administrators to be able to provide a network faster and without manual configuration. SDN can separate the control plane and data plane, administrators do not need to configure the existing network devices one by one [1]. With SDN networking, administrators gain independent control of the entire network at one point, which greatly simplifies network operations [2].

However, in addition to the many advantages that exist on this network, it turns out that the SDN network also has several issues, one of which is security. The higher the traffic, the higher the possibility of security threats being exploited by perpetrators for personal gain [3]. On this network, one of the most common attacks is Denial of Service (DoS). This attack is mostly carried out at layer 3 (application layer) and layer 4 (transport layer). DDoS is a type of attack that utilizes many IPs to carry out attacks on the network, namely by sending request packets that very much make traffic congestion on a network so that the work of the device becomes louder than usual, this results in damage to network devices, which is one of the main driving factors for attackers [4].

[*] **Corresponding author**. *E-mail address: irmairhamna@unpak.ac.id*

Received: 2 February 2022, Accepted: 14 July 2022, and available online 30 July 2022
DOI: https://doi.org/10.33751/komputasi.v19i2.4769

In 2016 [5] it was found that the percentage of DoS attacks was still quite large, 46.9% for layers 3 and 4. One of the preventions of this attack issue, some of which are by using a firewall [6], but by using this firewall attacks can be detected only 40% only. Therefore, this study uses machine learning as attack detection. Machine learning can help define rules on the SDN controller and can predict attacks more accurately[7].

Therefore we need a network analysis that aims to determine the attacks that appear on a computer network. In this study, the process of detecting attacks on the SDN network will be carried out. The SDN network will be built using mininet software, then the process of retrieving traffic data on the SDN network using wireshark. This study uses the NSL KDD dataset as training data and test data used to determine the algorithm that can be used or implemented on the controller.

2. Methods

2.1 Software Defined Network (SDN)

According to the Open Networking Foundation (ONF), SDN is defined as a network architecture that is able to separate control and forwarding functions to enable direct programmable network control and underlying infrastructure for network applications and services [9]. Software defined network (SDN) is a new concept in designing, managing and implementing networks. Increasingly complex needs and innovations make SDN present to support all needs and innovations in the network sector. The following is the SDN network architecture shown in Figure 1

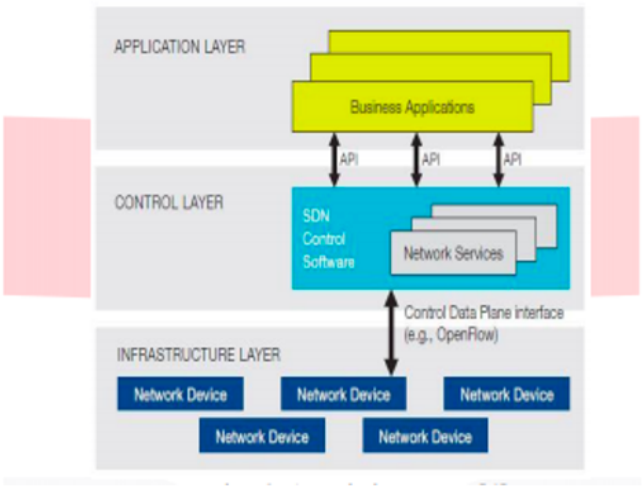


Figure 1. SDN Architecture.

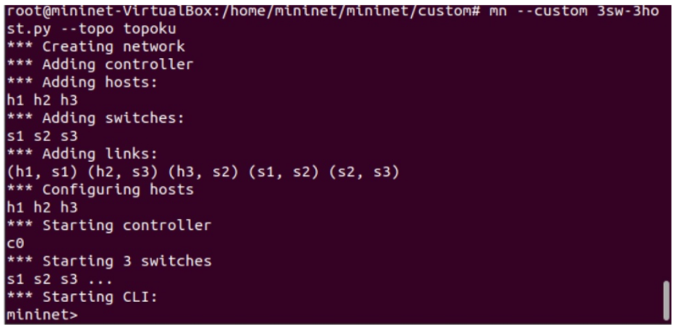
In the SDN architecture, the network is divided into 3 layers, namely the application layer, control layer and infrastructure/data layer. The application layer is the user interface in managing or developing an SDN network. Control plane is a controller that is centralized and based on software. Subordinate hardware is fully controlled by the controlling plane or controller in carrying out forwarding decisions. All subordinates are connected to the controller [8].

Most SDN network services are centralized in SDN control software. The SDN controller manages all network devices at the base infrastructure layer, which is a single logical virtual switch. The network operator can control the network via standard interfaces (eg, OpenFlow) independently of the network device vendor. The network devices are only equipped with the data forwarding function which is controlled by the SDN controller. This makes network design and operation simpler and more efficient. The API between the application layer and the control layer can provide virtualization of the network environment and the means to implement various policies regarding routing, access control, traffic engineering, management and so on [9].

2.2 Mininet

Mininet is one of the emulators that is widely used in building an SDN architecture. Mininet is a type of CLI emulator that is used to create a virtual network topology on a software defined network [10]. In the mininet, there are already several examples of topologies provided by the mininet, some of which are minimal, single, reversed, linear, and tree. Creating a topology manually can be done in mininet by entering a few lines of python code [11]. Mininet is a network emulator that creates a network of virtual hosts, switches, controllers, and links. The Mininet host runs standard Linux networking software, and its pocket-switch supports OpenFlow for highly flexible custom routing and Software-Defined Networking [12].

At this stage, the SDN architecture was built using a virtual box and mininet software. The network architecture at the SDN in this study is to create a virtual network that will be used to retrieve network traffic data. This study builds 1 ryu controller, with 3 switches and 3 hosts as shown in Figure 2 and assigning IP Addresses.



```
root@mininet-VirtualBox:/home/mininet/mininet/custom# mn --custom 3sw-3ho
st.py --topo topoku
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s3) (h3, s2) (s1, s2) (s2, s3)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet>
```

Figure 2. SDN Architecture.

The following is the assignment of IP Address for each host shown in Table 1.

Table 1. Host IP Sharing.

Host	IP Address
Host 1	192.168.1.1
Host 2	192.168.1.2
Host 3	192.168.1.3

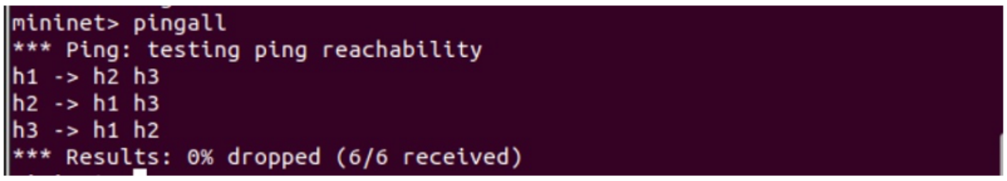
2.3 Wireshark

Wireshark Network Protocol Analyzer is a software application that is used to view and attempt to capture network packets and attempt to display all the information in the packets as detailed as possible. Open Source from Wireshark uses a Graphical User Interface (GUI) [13]. Wireshark is a free and open-source packet analyzer. These tools are often used to find problems in networks, software development and communication protocols, and education. Wireshark is cross-platform and uses pcap to capture network packets. Wireshark can run on almost all available operating systems [14]. Wireshark formerly known as Ethereum is a GUI based software protocol for network traffic. Wireshark allows users to interactively browse packets from a busy computer network or previously capture files [15].

2.4 Network attacks

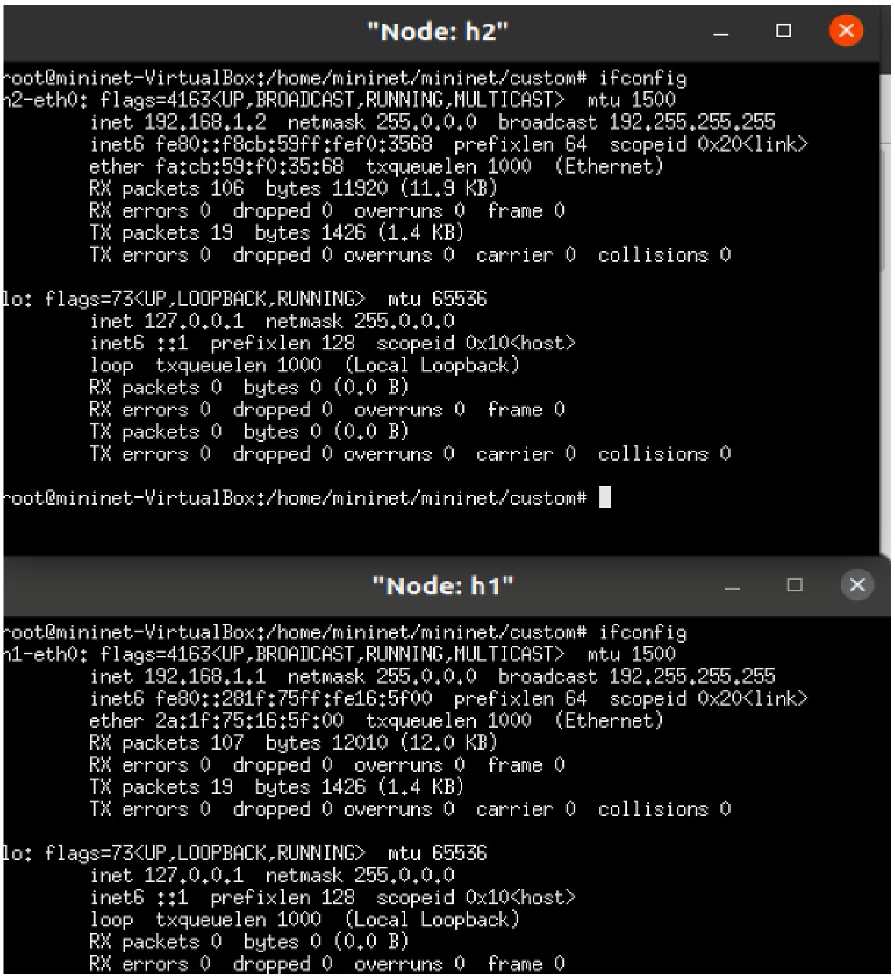
In general, the most common attacks on the internet network are flooding/denial of service (DOS). DDOS is a large-scale data transmission that is intentionally done to reduce router performance in data transmission media. Flooding is more often used at the data link layer, sent from the physical layer to the data link and recording MAC addresses in data compression is carried out continuously without going to the next layer. With flooding very detrimental to bandwidth and other users, it is necessary to have a flooding analysis process in order to know the characteristics of flooding data if it occurs on a router [16]. One type of attack on Dos is Smurf, this type is often carried out by flooding the victim with "echo-reply" Internet Control Message Protocol (ICMP) packets or commonly known as ping of death. In this condition the attacker will send many ICMP packets to the victim's broadcast address. The packets sent are the victim's address as the source IP address [17].

At this stage, it is doing a network simulation based on the SDN architecture that has been created with the aim of communicating between users. At this stage, simulation is carried out on the network architecture that has been built shown on Figure 3 and 4.



```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```

Figure 3. Connection test between hosts.



```
oot@mininet-VirtualBox:/home/mininet/mininet/custom# ifconfig
2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.2  netmask 255.0.0.0  broadcast 192.255.255.255
        inet6 fe80::f8cb:59ff:fef0:3568  prefixlen 64  scopeid 0x20<link>
        ether fa:cb:59:f0:35:68  txqueuelen 1000  (Ethernet)
        RX packets 106  bytes 11920 (11.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 19  bytes 1426 (1.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

oot@mininet-VirtualBox:/home/mininet/mininet/custom#
```

```
oot@mininet-VirtualBox:/home/mininet/mininet/custom# ifconfig
1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.1  netmask 255.0.0.0  broadcast 192.255.255.255
        inet6 fe80::281f:75ff:fe16:5f00  prefixlen 64  scopeid 0x20<link>
        ether 2a:1f:75:16:5f:00  txqueuelen 1000  (Ethernet)
        RX packets 107  bytes 12010 (12.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 19  bytes 1426 (1.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
```

Figure 4. Host node configuration

2.5 Classification

Classification is a technique for grouping data based on certain attributes or variables which will later be used to accurately predict new data that does not yet have a group or class [18]. Support Vector Machine is a machine learning method that compares a selection of discrete-valued parameters called candidate sets. The SVM classification process has been introduced by Vapnik, Boser, and Guyon [19]. Research conducted by R. Kokila et al that used SVM for the attack detection process was able to produce an accuracy of 95.11% [20]. Support Vector Machine is a machine learning method that compares a selection of discrete-valued parameters called candidate sets. The SVM classification process has been introduced by Vapnik, Boser, and Guyon [19]. Research conducted by R. Kokila et al that used SVM for the attack detection process was able to produce an accuracy of 95.11% [20].

After analyzing the traffic data in the simulation, the next step is to test the training data and test data taken from NSL KDD using data mining techniques. This dataset is a data set that is used to find out which algorithm has high accuracy. The results of this data are used as a reference to be implemented on a controller capable of detecting attacks on the SDN network. The NSL-KDD dataset has 125973 rows of data and 42 attributes. This stage is carried out, namely checking data packets indicated as attacks. After the traffic data is taken, the data will be compared during normal conditions and an attack occurs. In this study, classification was also carried out using the NSL KDD dataset. With this dataset, a classification process is carried out to find out which algorithms can be used or implemented into the SDN controller. In this process the data that occurs in the attack will be classified using the Weka software. The output of this weka will be visible or can detect how much data is considered to be entered as attack data.

3. Result and Discussion

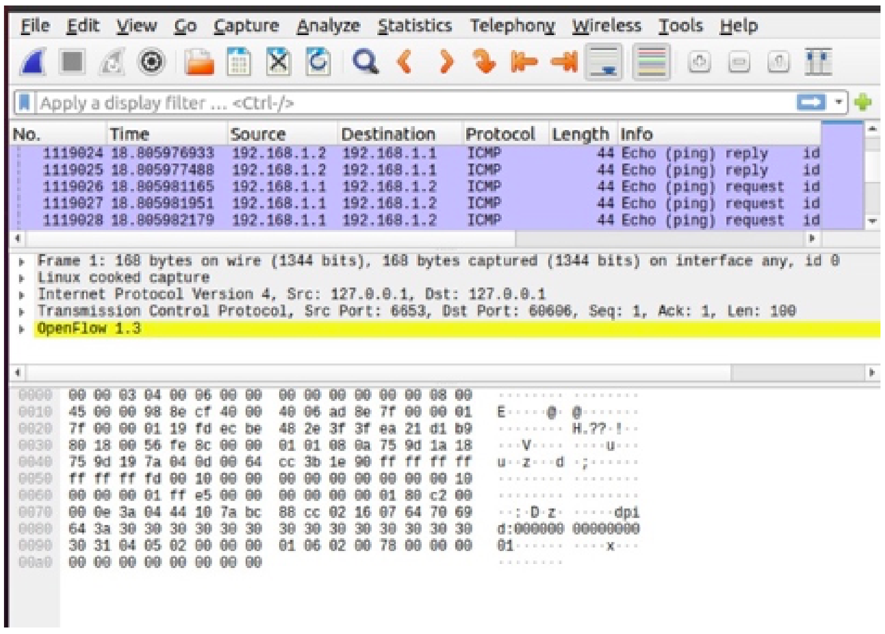
The process research that has been carried out includes building an SDN architecture, simulating attacks, retrieving data and performing attack detection. This study builds a virtual network on the SDN network. Traffic data will be taken from the virtual network with two conditions, namely normal network conditions and network attacks detected. This traffic data will be used as an analysis to determine the traffic comparison that occurs. The following is a description of the results of network traffic data collection, which are data attributes for the analysis process

1. Number, which is a sequence of data packet numbers captured by wireshark.
2. Time, which is the time when the packet that goes to the accessed website page is captured.
3. Source is the source ip of the packet, where the IP matches.
4. Destination is the destination IP of the packet.
5. Protocol is a display of what protocol the above data packet uses, namely HTTP.
6. Length is the length of data packet transmission to the destination address.
7. Info is a display of detailed information about the package mentioned above.

In this research, the data is filtered first, namely the data attributes in the form of numbers and info will not be included in the process of analyzing this data.

3.1 Normal Traffic Data

The following are the results of data traffic under normal conditions or without any attacks being carried out. Based on the data attributes, analysis can be carried out as shown in Figure 5 below Under normal conditions, the computer does not experience anything, and it can be seen from the data traffic that there are no anomalies or data changes.

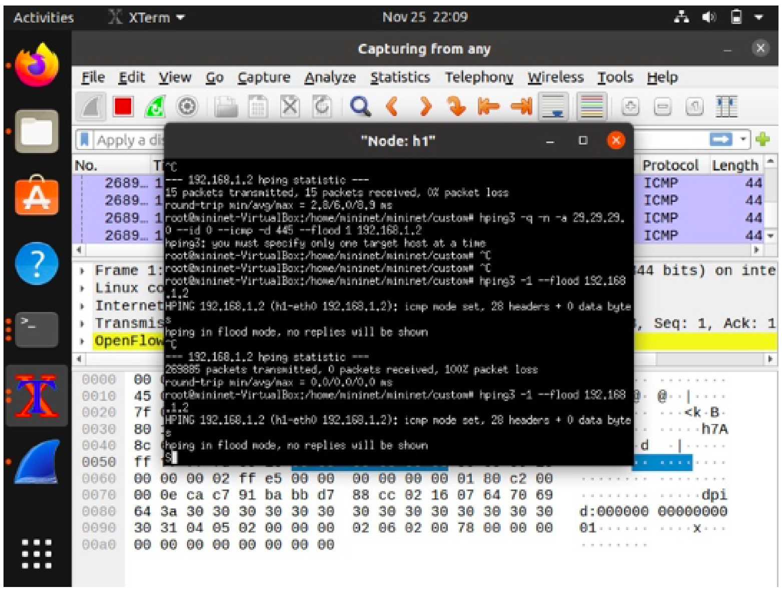


```
No.     Time         Source       Destination  Protocol Length Info
1119024 18.805976933 192.168.1.2  192.168.1.1  ICMP         44 Echo (ping) reply   id
1119025 18.805977488 192.168.1.2  192.168.1.1  ICMP         44 Echo (ping) reply   id
1119026 18.805981165 192.168.1.1  192.168.1.2  ICMP         44 Echo (ping) request id
1119027 18.805981951 192.168.1.1  192.168.1.2  ICMP         44 Echo (ping) request id
1119028 18.805982179 192.168.1.1  192.168.1.2  ICMP         44 Echo (ping) request id

Frame 1: 168 bytes on wire (1344 bits), 168 bytes captured (1344 bits) on interface any, id 0
Linux cooked capture
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 6653, Dst Port: 60606, Seq: 1, Ack: 1, Len: 100
OpenFlow 1.3

0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ········ ········
0010  45 00 00 98 8e cf 40 00  40 06 ad 8e 7f 00 00 01   E·····@· @·······
0020  7f 00 00 01 19 fd ec be  48 2e 3f 3f ea 21 d1 b9   ········ H.??·!··
0030  80 18 00 56 fe 8c 00 00  01 01 08 0a 75 9d 1a 18   ···V···· ····u···
0040  75 9d 19 7a 04 0d 00 64  cc 3b 1e 90 ff ff ff ff   u··z···d ·;······
0050  ff ff ff fd 00 10 00 00  00 00 00 00 00 00 00 10   ········ ········
0060  00 00 00 01 ff e5 00 00  00 00 00 00 01 80 c2 00   ········ ········
0070  00 0e 3a 04 44 10 7a bc  88 cc 02 16 07 64 70 69   ··:·D·z· ·····dpi
0080  64 3a 30 30 30 30 30 30  30 30 30 30 30 30 30 30   d:000000 00000000
0090  30 31 04 05 02 00 00 00  01 06 02 00 78 00 00 00   01······ ····x···
00a0  00 00 00 00 00 00 00 00                            ········
```

Figure 5. Traffic results using wireshark.

3.2 Traffic data by attack

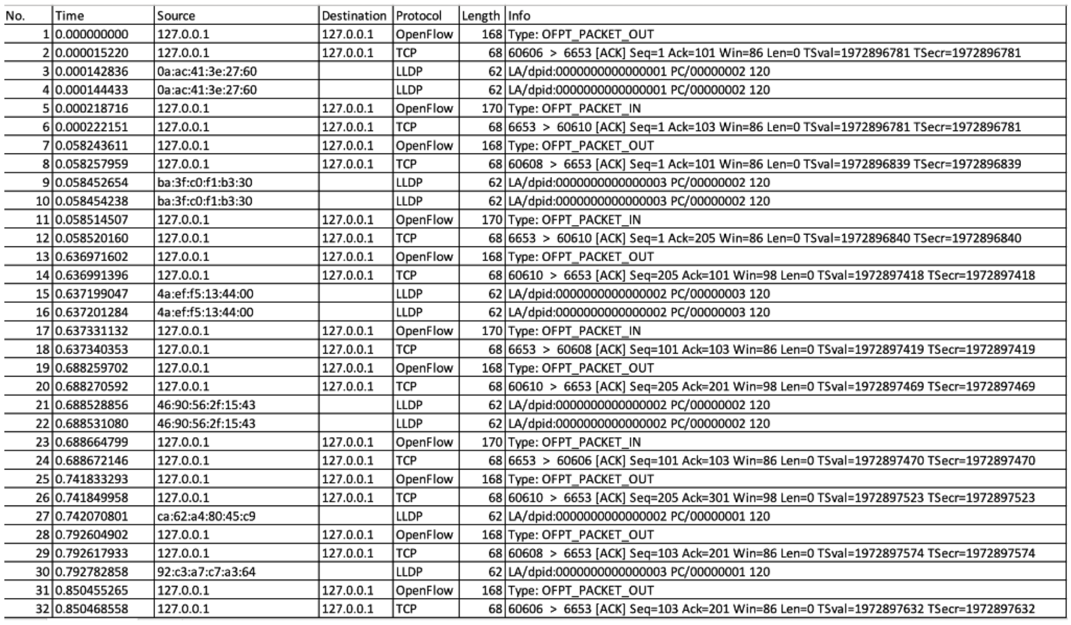
To retrieve traffic data on network conditions given an attack, the virtual mininet network node host1 will be treated as an attacker to host2.In this process, host1 will attack host2 using Ddos and SYN flooding. When attacked, host2 will experience flooding and host2 will hang as shown in Figure 6. This is because the type of DDos attack is a connectionless type which can deplete the network resources of the victim by continuously sending large amounts of data packets. In this condition, traffic data will be taken using wireshark as shown in Figure 7.



```
^C
--- 192.168.1.2 hping statistic ---
15 packets transmitted, 15 packets received, 0% packet loss
round-trip min/avg/max = 2.8/6.0/8.9 ms
root@mininet-VirtualBox:/home/mininet/mininet/custom# hping3 -q -n -a 29.29.29.
0 --id 0 --icmp -d 445 --flood 1 192.168.1.2
hping3: you must specify only one target host at a time
root@mininet-VirtualBox:/home/mininet/mininet/custom# ^C
root@mininet-VirtualBox:/home/mininet/mininet/custom# ^C
root@mininet-VirtualBox:/home/mininet/mininet/custom# hping3 -1 --flood 192.168
.1.2
HPING 192.168.1.2 (h1-eth0 192.168.1.2): icmp mode set, 28 headers + 0 data byte
s
hping in flood mode, no replies will be shown
^C
--- 192.168.1.2 hping statistic ---
269885 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@mininet-VirtualBox:/home/mininet/mininet/custom# hping3 -1 --flood 192.168
.1.2
HPING 192.168.1.2 (h1-eth0 192.168.1.2): icmp mode set, 28 headers + 0 data byte
s
hping in flood mode, no replies will be shown
$
```

Figure 6. Host1 attacks host 2.

In the attack scenario from host1 to host2, the traffic data retrieval process will be carried out using wireshark and will be carried out with the same process, namely the traffic data



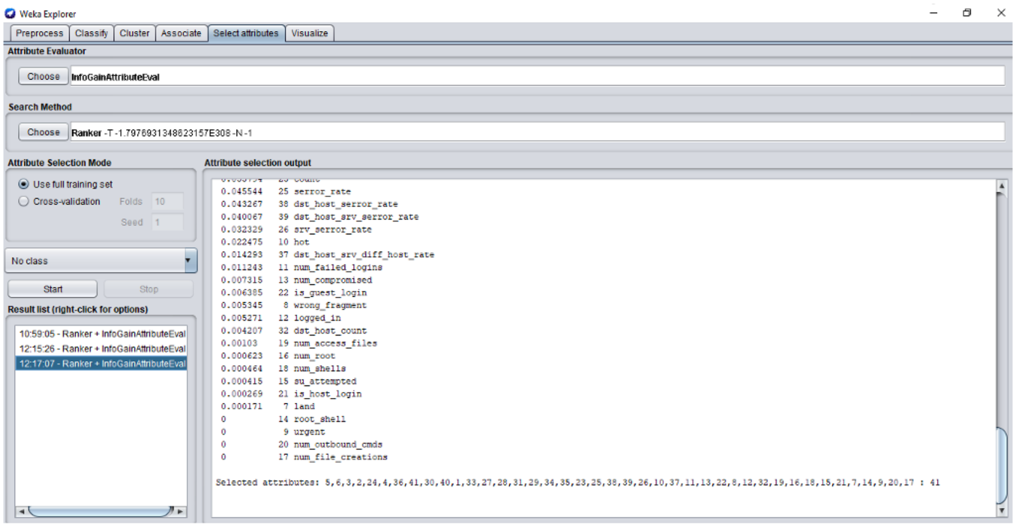
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	OpenFlow	168	Type: OFPT_PACKET_OUT
2	0.000015220	127.0.0.1	127.0.0.1	TCP	68	60606 > 6653 [ACK] Seq=1 Ack=101 Win=86 Len=0 TSval=1972896781 TSecr=1972896781
3	0.000142836	0a:ac:41:3e:27:60		LLDP	62	LA/dpid:0000000000000001 PC/00000002 120
4	0.000144433	0a:ac:41:3e:27:60		LLDP	62	LA/dpid:0000000000000001 PC/00000002 120
5	0.000218716	127.0.0.1	127.0.0.1	OpenFlow	170	Type: OFPT_PACKET_IN
6	0.000222151	127.0.0.1	127.0.0.1	TCP	68	6653 > 60610 [ACK] Seq=1 Ack=103 Win=86 Len=0 TSval=1972896781 TSecr=1972896781
7	0.058243611	127.0.0.1	127.0.0.1	OpenFlow	168	Type: OFPT_PACKET_OUT
8	0.058257959	127.0.0.1	127.0.0.1	TCP	68	60608 > 6653 [ACK] Seq=1 Ack=101 Win=86 Len=0 TSval=1972896839 TSecr=1972896839
9	0.058452654	ba:3f:c0:f1:b3:30		LLDP	62	LA/dpid:0000000000000003 PC/00000002 120
10	0.058454238	ba:3f:c0:f1:b3:30		LLDP	62	LA/dpid:0000000000000003 PC/00000002 120
11	0.058514507	127.0.0.1	127.0.0.1	OpenFlow	170	Type: OFPT_PACKET_IN
12	0.058520160	127.0.0.1	127.0.0.1	TCP	68	6653 > 60610 [ACK] Seq=1 Ack=205 Win=86 Len=0 TSval=1972896840 TSecr=1972896840
13	0.636971602	127.0.0.1	127.0.0.1	OpenFlow	168	Type: OFPT_PACKET_OUT
14	0.636991396	127.0.0.1	127.0.0.1	TCP	68	60610 > 6653 [ACK] Seq=205 Ack=101 Win=98 Len=0 TSval=1972897418 TSecr=1972897418
15	0.637199047	4a:ef:f5:13:44:00		LLDP	62	LA/dpid:0000000000000002 PC/00000003 120
16	0.637201284	4a:ef:f5:13:44:00		LLDP	62	LA/dpid:0000000000000002 PC/00000003 120
17	0.637331132	127.0.0.1	127.0.0.1	OpenFlow	170	Type: OFPT_PACKET_IN
18	0.637340353	127.0.0.1	127.0.0.1	TCP	68	6653 > 60608 [ACK] Seq=101 Ack=103 Win=86 Len=0 TSval=1972897419 TSecr=1972897419
19	0.688259702	127.0.0.1	127.0.0.1	OpenFlow	168	Type: OFPT_PACKET_OUT
20	0.688270592	127.0.0.1	127.0.0.1	TCP	68	60610 > 6653 [ACK] Seq=205 Ack=201 Win=98 Len=0 TSval=1972897469 TSecr=1972897469
21	0.688528856	46:90:56:2f:15:43		LLDP	62	LA/dpid:0000000000000002 PC/00000002 120
22	0.688531080	46:90:56:2f:15:43		LLDP	62	LA/dpid:0000000000000002 PC/00000002 120
23	0.688664799	127.0.0.1	127.0.0.1	OpenFlow	170	Type: OFPT_PACKET_IN
24	0.688672146	127.0.0.1	127.0.0.1	TCP	68	6653 > 60606 [ACK] Seq=101 Ack=103 Win=86 Len=0 TSval=1972897470 TSecr=1972897470
25	0.741833293	127.0.0.1	127.0.0.1	OpenFlow	168	Type: OFPT_PACKET_OUT
26	0.741849958	127.0.0.1	127.0.0.1	TCP	68	60610 > 6653 [ACK] Seq=205 Ack=301 Win=98 Len=0 TSval=1972897523 TSecr=1972897523
27	0.742070801	ca:62:a4:80:45:c9		LLDP	62	LA/dpid:0000000000000002 PC/00000001 120
28	0.792604902	127.0.0.1	127.0.0.1	OpenFlow	168	Type: OFPT_PACKET_OUT
29	0.792617933	127.0.0.1	127.0.0.1	TCP	68	60608 > 6653 [ACK] Seq=103 Ack=201 Win=86 Len=0 TSval=1972897574 TSecr=1972897574
30	0.792782858	92:c3:a7:c7:a3:64		LLDP	62	LA/dpid:0000000000000003 PC/00000001 120
31	0.850455265	127.0.0.1	127.0.0.1	OpenFlow	168	Type: OFPT_PACKET_OUT
32	0.850468558	127.0.0.1	127.0.0.1	TCP	68	60606 > 6653 [ACK] Seq=103 Ack=201 Win=86 Len=0 TSval=1972897632 TSecr=1972897632

Figure 7. Traffic data collection.

will be processed via Weka. The results of the traffic analysis on Weka are to determine the distribution of the protocol, destination IP and source in the data.

Next is to process the dataset from NSL KDD. This dataset has 42 attributes shown in Table 1 and with 2 classes, namely normal and anomaly. In this process, attribute selection is first performed using information gain. Information Gain is a feature selection method that can generate attribute rankings.

From the results of this attribute selection, there are 4 attributes that will be removed because there are 0 values, namely root_shel, urgent, numb_outbond_cmds, num_file_creations which are shown in Figure 8. So there are 38 attributes used in this study.



Attribute Evaluator: InfoGainAttributeEval

Search Method: Ranker -T -1.7976931348623157E308 -N -1

Attribute Selection Mode: Use full training set; Cross-validation Folds 10 Seed 1

No class

Result list (right-click for options)
10:59:05 - Ranker + InfoGainAttributeEval
12:15:26 - Ranker + InfoGainAttributeEval
12:17:07 - Ranker + InfoGainAttributeEval

Attribute selection output

```
 0.045544   25 serror_rate
 0.043267   38 dst_host_serror_rate
 0.040067   39 dst_host_srv_serror_rate
 0.032329   26 srv_serror_rate
 0.022475   10 hot
 0.014293   37 dst_host_srv_diff_host_rate
 0.011243   11 num_failed_logins
 0.007315   13 num_compromised
 0.006385   22 is_guest_login
 0.005345    8 wrong_fragment
 0.005271   12 logged_in
 0.004207   32 dst_host_count
 0.00103    19 num_access_files
 0.000623   16 num_root
 0.000464   18 num_shells
 0.000415   15 su_attempted
 0.000269   21 is_host_login
 0.000171    7 land
 0          14 root_shell
 0           9 urgent
 0          20 num_outbound_cmds
 0          17 num_file_creations

Selected attributes: 5,6,3,2,24,4,36,41,30,40,1,33,27,28,31,29,34,35,23,25,38,39,26,10,37,11,13,22,8,12,32,19,16,18,15,21,7,14,9,20,17 : 41
```

Figure 8. Feature selection results.

Next process is the classification process using machine learning. The research has 2 objective

Table 2. Attributes of NSL KDD.

No	Atribute	No	Attribute
1	Duration	24	erv_count
2	Protocol type	25	Serror_rate
3	Service	26	Srv_serror_rate
4	Flag	27	rerror_rate
5	Scr_byte	28	Srv_serror_rate
6	Dst_byte	29	Same_srv_rate
7	Land	30	Diff_serv_rate
8	woring_fragment	31	Srv_diff_rate
9	Urgent	32	Dst_host_count
10	Host	33	Dst_host_srv_count
11	Numb_failed_logins	34	Dst_host_srv_host
12	Logged_in	35	Dst_host_diff_srv_host
13	Num_compromised	36	Dst_host_same_srv_port _rate
14	Logged_in	37	Dst_host_srv_diff_host _rate
15	Su_attempted	38	Dst_host_serror_rate
16	Num_root	39	Dst_host_srv_serror_rate
17	Num_file_creation	40	Dst_host_serror_rate
18	Num_shell	41	Dst_host_srv_serror_rate
19	Num_access_file		
20	Num_outbondcmds		
21	Is_hott_login		
22	Is_guest_login		
23	Count		

classes, namely normal and anomaly and the K fold value used is 10. This dataset is carried out using several algorithms and is generated through the parameters of accuracy, precision, recal, f-measure and time taken building which are shown in Table 3.

Table 3. The results of the dataset classification trial.

Algorithm	Accuracy(%)	Precision (%)	Recall (%)	F Measure (%)	Time taken building (Detik)
SVM	91.8	91,7	91.8	91.2	14.64
C4.5	97.08	97,1	97.1	97.1	0.45
Random Forest	97.7	97,7	97.7	97.7	1.49
Naive bayes	65.6	83,1	65.7	69.7	0.17

4. Conclusion

The research succeeded in building a network simulation using SDN using Ryu Controller and mininet. Based on the results of research that has been carried out that Ddos attacks on SDN networks work by sending data packets in very large numbers, causing computer performance to be disrupted, as well as reducing computer resources. From the results of testing the dataset from NSL KDD using Random Forest, the results obtained are 97.7% accuracy. From this accuracy value, it is stated that the algorithm can be used to detect SDN attacks to be implemented in network simulations.

References

[1] R. Kandoi and M. Antikainen, “Denial-of-service attacks in openflow sdn networks,” in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, May 2015, pp. 1322–1326.

[2] S. Scott-Hayward, S. Natarajan, and S. Sezer, “A survey of security in software defined networks,” *IEEE Communications Surveys Tutorials,* vol. 18, no. 1, pp. 623–654, Firstquarter 2016.

[3] R. F. Pratama, Perancangan dan Implementasi Adaptive Intrusion Prevention System (IPS) untuk Pencegahan Penyerangan pada Arsitektur Software-Defined Network (SDN), 2017.

[4] J. N. Bakker, “Intelligent Traffic Classification for Detecting DDoS Attacks using SDN/OpenFlow,” 2017.

[5] N. Guard. (2016, Jun). Ddos threat report reflection attacks: Q2 2016.

[6] A. El-Atawy, E. Al-Shaer, T. Tran, and R. Boutaba, “Adaptive early packet filtering for defending firewalls against dos attacks,” in I*EEE INFOCOM 2009*, April 2009, pp. 2437–2445.

[7] S. Nanda, F. Zafari, C. DeCusatis, E. Wedaa, and B. Yang, “Predicting network attack patterns in sdn using machine learning approach,” in 2016 *IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2016, pp. 167–172.

[8] O. N. F. W. Paper, "Software-Defined Networking: The New Norm for Networks," 2012.

[9] R.M Negara dan R. Tulloh, 2017. Analisis Simulasi Penerapan Algoritma OSPF Menggunakan RouteFlow pada Jaringan Software Defined Network (SDN) *Jurnal Infotel* Vol.9 No.1 Februari 2017 http://dx.doi.org/10.20895/infotel.v9i1.172 ISSN : 2085-3688; e-ISSN : 2460-0997.

[10] M, M Azis, Y Azhar , 2020. Saifuddin. Analisa Sistem Identifikasi DDoS Menggunakan KNN Pada Jaringan Software Defined Network(SDN). *REPOSITOR*, Vol. 2, No. 7, Juli 2020, Pp. 915-922 ISSN : 2714-7975 E-ISSN : 2716-1382.

[11] K. Kaur, J. Singh, and N. S. Ghumman, “Mininet as Software Defined Networking Testing Platform,” Int. *Conf. Commun. Comput. Syst.*, pp. 3–6, 2014.

[12] R Edgar , A.T,Hanuranto2 , O. Mentari, 2019. Perancangan dan Analisis Sistem pada Kontroler Pox, Ryu, dan Opendaylight Pada Software Defined Network Design And Analysis System On Controller Pox, Ryu, And Opendaylight On Software Defined Network ISSN : 2355-9365 *e-Proceeding of Engineering* : Vol.6, No.2 Agustus 2019. Page 4433.

[13] Sihombing, R.O.L, Zulfin,M . 2013. Analisis Kinerja Trafik Web Browser Dengan Wireshark Network Protocol Analyzer Pada Sistem Client-Server. *Singuda Ensikom* Vol. 2 No. 3 Juni 2013.

[14] Wulandari, R. 2016. Analisis Qos (Quality Of Service) Pada Jaringan Internet (Studi Kasus : Upt Loka Uji Teknik Penambangan Jampang Kulon – Lipi. *Jurnal Teknik Informatika dan Sistem Informasi,* Vol. 2, No. 2,pp. 162-172,Agustus 2016.

[15] Sujana, A.P. 2014. Perangkat Pendukung Forensik Lalu Lintas Jaringan. *Jurnal Teknik Komputer Unikom – Komputika* Vol. 3, No.1, 2014.

[16] Hendrawan, A.H. 2016. Analisis Serangan Flooding Data Pada Router Mikrotik. *Jurnal Kreatif.* Vol. 4 No. 1.

[17] Patrikakis, C., Masikos, M., and Zouraraki, O. Distributed Denial of Service Attacks. *The Internet Protocol Journal.* Volume 7, Number 4.

[18] Azis, M. M., Azhar, Y., dan Syaifuddin, S. (2020). Analisa Sistem Identifikasi DDoS Menggunakan KNN Pada Jaringan Software Defined Network (SDN). *Jurnal Repositor,* 2(7), 915-922.

[19] Riadi, I., Umar, R., dan Aini, F. D. (2019). Analisis Perbandingan Detection Traffic Anomaly Dengan Metode Naive Bayes Dan Support Vector Machine (Svm). *ILKOM Jurnal Ilmiah*, 11(1), 17-24.

[20] R. T. Kokila, S. Thamarai Selvi, and K. Govindarajan, "DDoS detection and analysis in SDN-based environment using support vector machine classifier," 6th Int. *Conf. Adv. Comput. ICoAC 2014*, pp. 205–210, 2015.

[21] Vembandasamy, K., Sasipriya, R., Deepa, E., 2015, Heart Diseases Detection Using Naive Bayes Algorithm, *International Journal of Innovative Science Engineering and Technology (IJISET)*, No. 9, Vol. 2, Hal. 441–444.

[22] Purushottam, Saxena, K., and Sharma, R. 2016. Efficient Heart Disease Prediction System using Decision Tree. International Conference on Computing, *Communication and Automation (ICCCA)*, Noida, India, 15-16 May. 72-77. DOI: 10.1109/CCAA.2015.7148346.

[23] Kurniabudi, K., Harris, A., dan Rahim, A. (2020). Seleksi Fitur Dengan Information Gain Untuk Meningkatkan Deteksi Serangan DDoS menggunakan Random Forest. *Techno. Com*, 19(1), 56-66.

[24] D. Summeet and D. Xian, *Data Mining and Machine Learning in Cybersecurity*. CRC Press, 2011.