DAFTAR PUSTAKA

Azodolmolky, Siamak. Software Defined Networking with OpenFlow. Birmingham: Packt Publishing, 2013

Rodríguez P., et al. Using Mininet for emulation and prototyping Software-Defined Networks, 2014.

Ahmad, I., et al. Security in Software Defined Networks: A Survey Controller. 2015

E.V Josy et al. Information and Communication Technology for Competitive Strategies ICTCS. 2020

Lorenzo Piccioni. Xterm : A Flexible Stkitard-Compliant XML-Based Termbase System. 2004

Ujjan, R. M. A., et al. Towards sFlow and adaptive polling sampling for deep learning based DDoS detection in SDN. 2020

Valdovinos, I. A., et al. Emerging DDoS attack detection and mitigation strategies in software-defined networks: Taxonomy, challenges and future directions. 2021

Abdullah, M., Al-awad, N., & Hussein, F. W. (2018). Performance Comparison and Evaluation of Different Software Defined Networks Controllers. International Journal of Computing and Network Technology, 6(2).

Maaloul, Rihab & Taktak, Raouia & Fourati, Lamia & Cousin, Bernard. (2018). Energy-Aware Routing in Carrier-Grade Ethernet using SDN Approach. IEEE Transactions on Green Communications and Networking. 2. 844-858. 10.1109/TGCN.2018.2832658.

Kalliola, Aapo & Lee, Kiryong & Lee, Heejo & Aura, Tuomas. (2015). Flooding DDoS mitigation and traffic management with software defined networking. 248-254. 10.1109/CloudNet.2015.7335317.

R. Klöti, V. Kotronis and P. Smith, "OpenFlow: A security analysis," *2013 21st IEEE International Conference on Network Protocols (ICNP)*,

LAMPIRAN

Lampiran 1 : Source Code

sFlow agent script

```
//this code for final project purposes so read it carefully
var ryu = '127.0.0.1';
var controls = {};

setFlow('udp_reflection',
{keys:'ipdestination,udpsourceport',value:'frames'});
setThreshold('udp_reflection_attack',
{metric:'udp_reflection',value:100,byFlow:true,timeout:2});

setEventHandler(function(evt) {
// don't consider inter-switch links
var link = topologyInterfaceToLink(evt.agent,evt.dataSource);
if(link) return;

// get port information
var port = topologyInterfaceToPort(evt.agent,evt.dataSource);
if(!port) return;

// need OpenFlow info to create Ryu filtering rule
if(!port.dpid || !port.ofport) return;

// we already have a control for this flow
if(controls[evt.flowKey]) return;

var [ipdestination,udpsourceport] = evt.flowKey.split(',');
var msg = {
priority:40000,
dpid:parseInt(port.dpid,16),
```

```
match: {
in_port:port.ofport,
dl_type:0x800,
nw_dst:ipdestination+'/32',
nw_proto:17,
tp_src:udpsourceport
}
};

var resp = http2({
url:'http://'+ryu+':8080/stats/flowentry/add',
headers:{'Content-Type':'application/json','Accept':'application/json'},
operation:'post',
body: JSON.stringify(msg)
});

controls[evt.flowKey] = {
time:Date.now(),
threshold:evt.thresholdID,
agent:evt.agent,
metric:evt.dataSource+'.'+evt.metric,
msg:msg
};

logInfo("blocking " + evt.flowKey);
},['udp_reflection_attack']);

setIntervalHandler(function() {
var now = Date.now();
for(var key in controls) {
```

```
let rec = controls[key];

// keep control for at least 10 seconds
if(now - rec.time < 10000) continue;
// keep control if threshold still triggered
if(thresholdTriggered(rec.threshold,rec.agent,rec.metric,key)) continue;

var resp = http2({
url:'http://'+ryu+':8080/stats/flowentry/delete',
headers:{'Content-Type':'application/json','Accept':'application/json'},
operation:'post',
body: JSON.stringify(rec.msg)
});

delete controls[key];

logInfo("unblocking " + key);
}
});
```

Ryu Controller script

```
from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import CONFIG_DISPATCHER,
MAIN_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.ofproto import ofproto_v1_3
from ryu.lib.packet import packet
```

```
from ryu.lib.packet import ethernet
from ryu.lib.packet import ether_types

class SimpleSwitch13(app_manager.RyuApp):
OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

def __init__(self, *args, **kwargs):
super(SimpleSwitch13, self).__init__(*args, **kwargs)
self.mac_to_port = {}

@set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
def switch_features_handler(self, ev):
datapath = ev.msg.datapath
ofproto = datapath.ofproto
parser = datapath.ofproto_parser

# install table-miss flow entry
#
# We specify NO BUFFER to max_len of the output action due to
# OVS bug. At this moment, if we specify a lesser number, e.g.,
# 128, OVS will send Packet-In with invalid buffer_id and
# truncated packet data. In that case, we cannot output packets
# correctly. The bug has been fixed in OVS v2.1.0.
match = parser.OFPMatch()
actions = [parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,
ofproto.OFPCML_NO_BUFFER)]
self.add_flow(datapath, 0, match, actions)

def add_flow(self, datapath, priority, match, actions, buffer_id=None):
ofproto = datapath.ofproto
parser = datapath.ofproto_parser
```

```
inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
actions)]
if buffer_id:
mod = parser.OFPFlowMod(datapath=datapath, buffer_id=buffer_id,
priority=priority, match=match,
instructions=inst)
else:
mod = parser.OFPFlowMod(datapath=datapath, priority=priority,
match=match, instructions=inst)
datapath.send_msg(mod)

@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def _packet_in_handler(self, ev):
# If you hit this you might want to increase
# the "miss_send_length" of your switch
if ev.msg.msg_len < ev.msg.total_len:
self.logger.debug("packet truncated: only %s of %s bytes",
ev.msg.msg_len, ev.msg.total_len)
msg = ev.msg
datapath = msg.datapath
ofproto = datapath.ofproto
parser = datapath.ofproto_parser
in_port = msg.match['in_port']

pkt = packet.Packet(msg.data)
eth = pkt.get_protocols(ethernet.ethernet)[0]

if eth.ethertype == ether_types.ETH_TYPE_LLDP:
# ignore lldp packet
```

```
        return
        dst = eth.dst
        src = eth.src

        dpid = format(datapath.id, "d").zfill(16)
        self.mac_to_port.setdefault(dpid, {})

        self.logger.info("packet in %s %s %s %s", dpid, src, dst, in_port)

        # learn a mac address to avoid FLOOD next time.
        self.mac_to_port[dpid][src] = in_port

        if dst in self.mac_to_port[dpid]:
            out_port = self.mac_to_port[dpid][dst]
        else:
            out_port = ofproto.OFPP_FLOOD

        actions = [parser.OFPActionOutput(out_port)]

        # install a flow to avoid packet_in next time
        if out_port != ofproto.OFPP_FLOOD:
            match = parser.OFPMatch(in_port=in_port, eth_dst=dst, eth_src=src)
            # verify if we have a valid buffer_id, if yes avoid to send both
            # flow_mod & packet_out
            if msg.buffer_id != ofproto.OFP_NO_BUFFER:
                self.add_flow(datapath, 1, match, actions, msg.buffer_id)
                return
            else:
                self.add_flow(datapath, 1, match, actions)
        data = None
        if msg.buffer_id == ofproto.OFP_NO_BUFFER:
```

```
data = msg.data

out = parser.OFPPacketOut(datapath=datapath, buffer_id=msg.buffer_id,
in_port=in_port, actions=actions, data=data)
datapath.send_msg(out)
```

Mininet Script

```
from mininet.node import CPULimitedHost
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.log import setLogLevel, info
from mininet.node import RemoteController
from mininet.cli import CLI
"""
Instructions to run the topo:
1. Go to directory where this fil is.
2. run: sudo -E python Simple_Pkt_Topo.py.py
The topo has 4 switches and 4 hosts. They are connected in a star shape.
"""

class SimplePktSwitch(Topo):
"""Simple topology example."""

def __init__(self, **opts):
"""Create custom topo."""

# Initialize topology
# It uses the constructor for the Topo cloass
super(SimplePktSwitch, self).__init__(**opts)

# Add hosts and switches
h1 = self.addHost('h1')
h2 = self.addHost('h2')
h3 = self.addHost('h3')
h4 = self.addHost('h4')

# Adding switches
```

```
s1 = self.addSwitch('s1', dpid="0000000000000001")
s2 = self.addSwitch('s2', dpid="0000000000000002")
s3 = self.addSwitch('s3', dpid="0000000000000003")
s4 = self.addSwitch('s4', dpid="0000000000000004")

# Add links
self.addLink(h1, s1)
self.addLink(h2, s2)
self.addLink(h3, s3)
self.addLink(h4, s4)

self.addLink(s1, s2)
self.addLink(s1, s3)
self.addLink(s1, s4)

def run():
c = RemoteController('c', '0.0.0.0', 6633)
net = Mininet(topo=SimplePktSwitch(), host=CPULimitedHost,
controller=None)
net.addController(c)
net.start()

CLI(net)
net.stop()

# if the script is run directly (sudo custom/optical.py):
if __name__ == '__main__':
setLogLevel('info')
run()
```

sFlow.py script

```
from mininet.net import Mininet
from mininet.log import info
from mininet.util import quietRun
from os import listdir, environ
from json import dumps
from re import match
from fcntl import ioctl
from array import array
from struct import pack, unpack
from sys import maxsize
import socket
import sys

try:
from urllib.request import build_opener, HTTPHandler, Request
except ImportError:
from urllib2 import build_opener, HTTPHandler, Request

def wrapper(fn):

def getIfInfo(dst):
is_64bits = maxsize > 2**32
struct_size = 40 if is_64bits else 32
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
max_possible = 8 # initial value
while True:
bytes = max_possible * struct_size
names = array('B')
for i in range(0, bytes):
```

```
names.append(0)
outbytes = unpack('iL', ioctl(
s.fileno(),
0x8912, # SIOCGIFCONF
pack('iL', bytes, names.buffer_info()[0])
))[0]
if outbytes == bytes:
max_possible *= 2
else:
break
s.connect((dst, 0))
ip = s.getsockname()[0]
for i in range(0, outbytes, struct_size):
addr = socket.inet_ntoa(names[i+20:i+24])
if addr == ip:
name = names[i:i+16]
try:
name = name.tobytes().decode('utf-8')
except AttributeError:
name = name.tostring()
name = name.split('\0', 1)[0]
return (name,addr)

def configSFlow(net,collector,ifname,sampling,polling):
info("*** Enabling sFlow:\n")
sflow = 'ovs-vsctl -- --id=@sflow create sflow agent=%s target=%s
sampling=%s polling=%s --' % (ifname,collector,sampling,polling)
for s in net.switches:
sflow += ' -- set bridge %s sflow=@sflow' % s
info(' '.join([s.name for s in net.switches]) + "\n")
```

```
quietRun(sflow)

def sendTopology(net,agent,collector):
info("*** Sending topology\n")
topo = {'nodes':{}, 'links':{}}
for s in net.switches:
topo['nodes'][s.name] = {'agent':agent, 'ports':{}}
path = '/sys/devices/virtual/net/'
for child in listdir(path):
parts = match('(^.+)-(.+)', child)
if parts == None: continue
if parts.group(1) in topo['nodes']:
ifindex = open(path+child+'/ifindex').read().split('\n',1)[0]
topo['nodes'][parts.group(1)]['ports'][child] = {'ifindex': ifindex}
i = 0
for s1 in net.switches:
j = 0
for s2 in net.switches:
if j > i:
intfs = s1.connectionsTo(s2)
for intf in intfs:
s1ifIdx = topo['nodes'][s1.name]['ports'][intf[0].name]['ifindex']
s2ifIdx = topo['nodes'][s2.name]['ports'][intf[1].name]['ifindex']
linkName = '%s-%s' % (s1.name, s2.name)
topo['links'][linkName] = {'node1': s1.name, 'port1': intf[0].name, 'node2':
s2.name, 'port2': intf[1].name}
j += 1
i += 1

opener = build_opener(HTTPHandler)
```

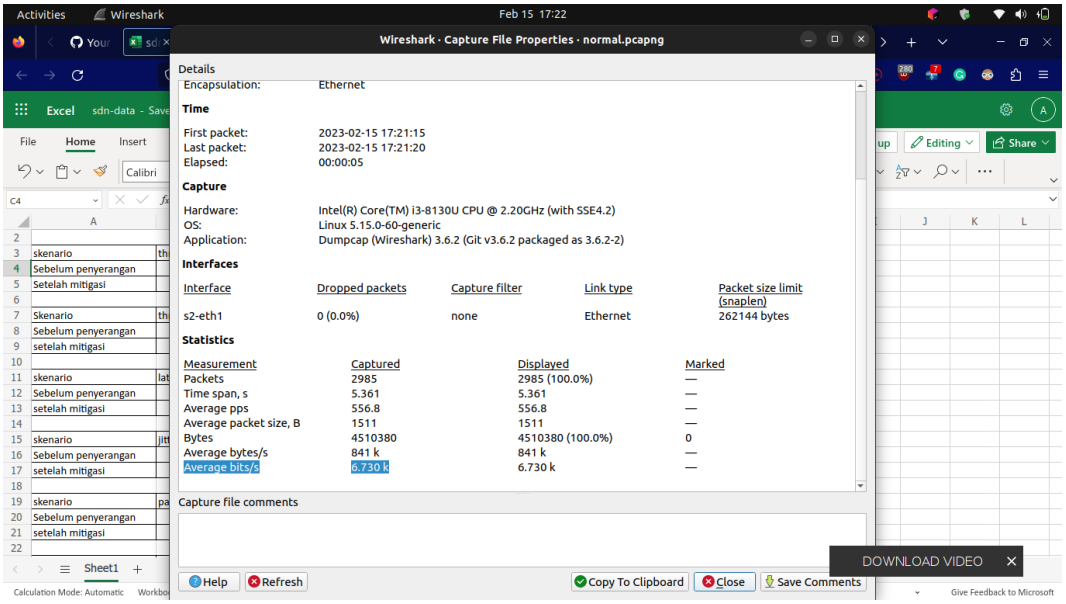
```
request = Request('http://%s:8008/topology/json' % collector,
data=dumps(topo).encode('utf-8'))
request.add_header('Content-Type','application/json')
request.get_method = lambda: 'PUT'
url = opener.open(request)

def result(*args,**kwargs):
res = fn(*args,**kwargs)
net = args[0]
collector = environ.get('COLLECTOR','127.0.0.1')
sampling = environ.get('SAMPLING','10')
polling = environ.get('POLLING','10')
(ifname, agent) = getIfInfo(collector)
configSFlow(net,collector,ifname,sampling,polling)
sendTopology(net,agent,collector)
return res

return result

setattr(Mininet, 'start', wrapper(Mininet.__dict__['start']))
```

start.sh

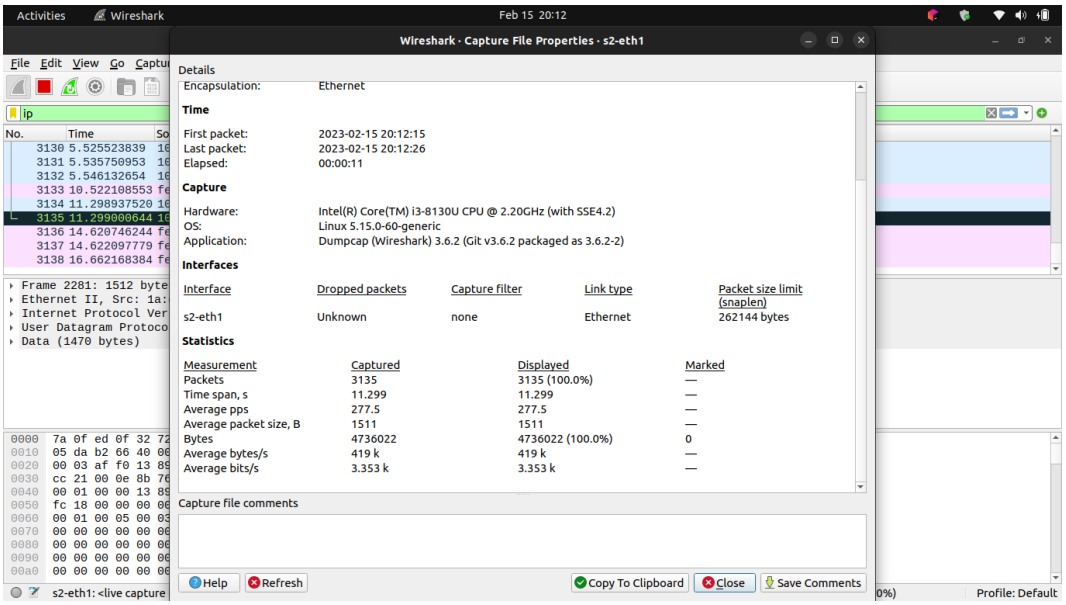
```
#!/bin/sh

HOME=`dirname $0`
cd $HOME

RTMEM="${RTMEM:-200M}"
RTMEMLIMITS="${RTMEMLIMITS:--Xms${RTMEM} -Xmx${RTMEM}}"
RTGC="${RTGC:--XX:+UseG1GC -XX:+UseStringDeduplication -
XX:MaxGCPauseMillis=100}"

JAR="./lib/sflowrt.jar"

exec java ${RTMEMLIMITS} ${RTGC} ${RTAPP} ${RTPROP} $@ -jar ${JAR}
```

```
#!/bin/sh

HOME=`dirname $0`
cd $HOME

RTMEM="${RTMEM:-200M}"
RTMEMLIMITS="${RTMEMLIMITS:--Xms${RTMEM} -Xmx${RTMEM}}"
RTGC="${RTGC:--XX:+UseG1GC -XX:+UseStringDeduplication -
XX:MaxGCPauseMillis=100}"

JAR="./lib/sflowrt.jar"

exec java ${RTMEMLIMITS} ${RTGC} ${RTAPP} ${RTPROP} $@ -jar ${JAR}
```

Lampiran 2 : Wireshark Capture

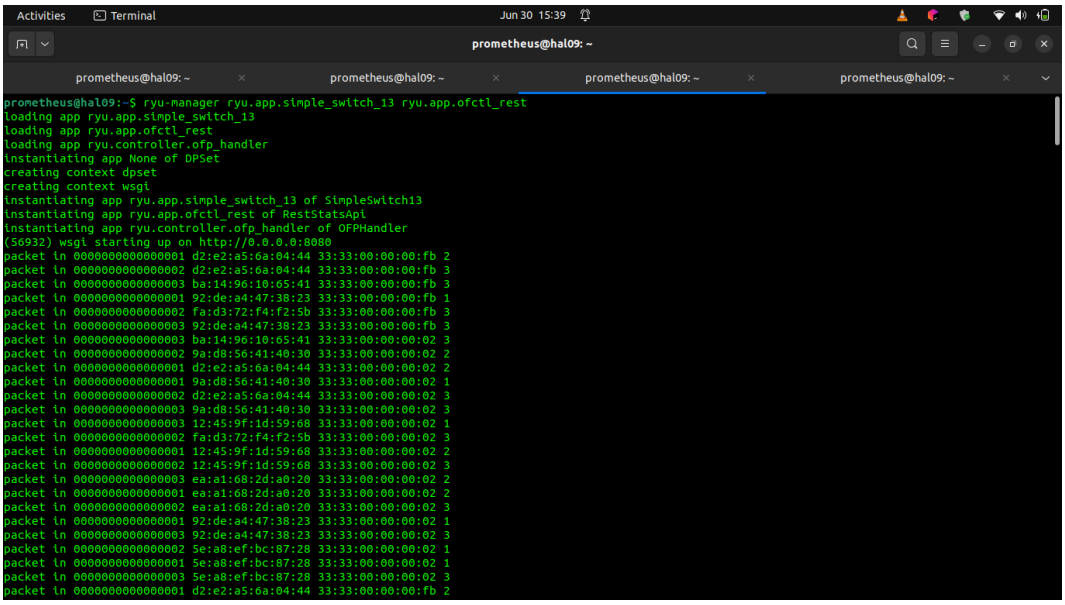


Lampiran 1: Skenario Normal

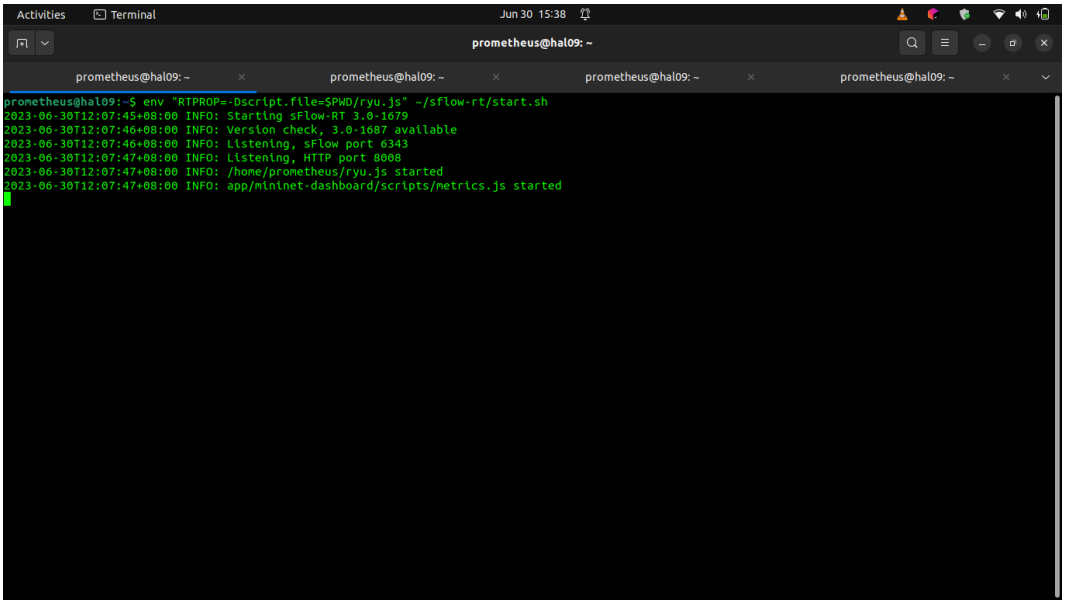


Lampiran 2: Skenario setelah mitigasi

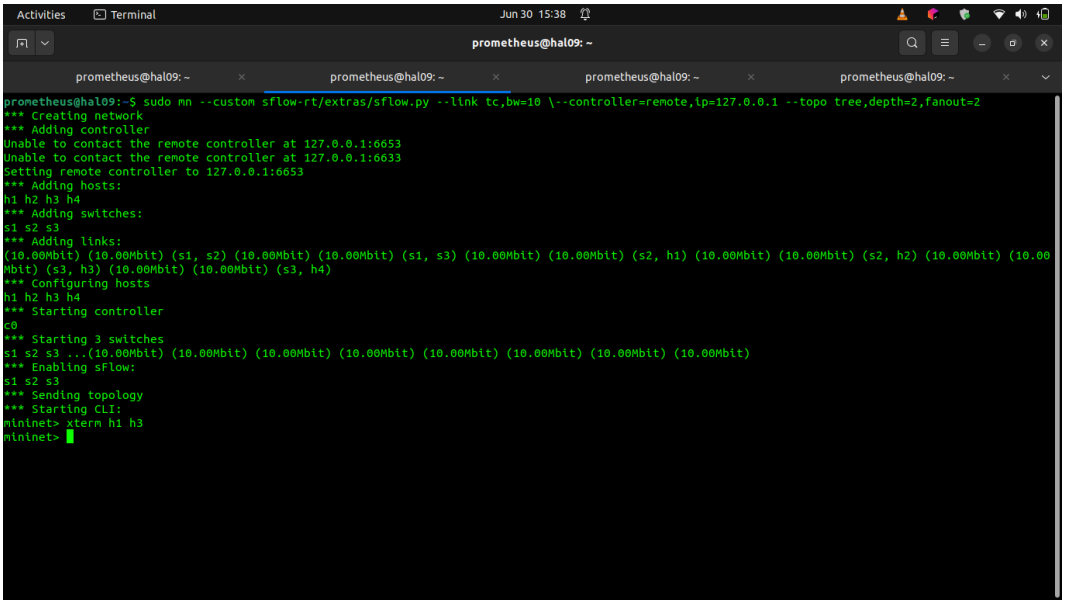
Lampiran 3 : Tampilan Sistem



Lampiran 3: Controller Ryu

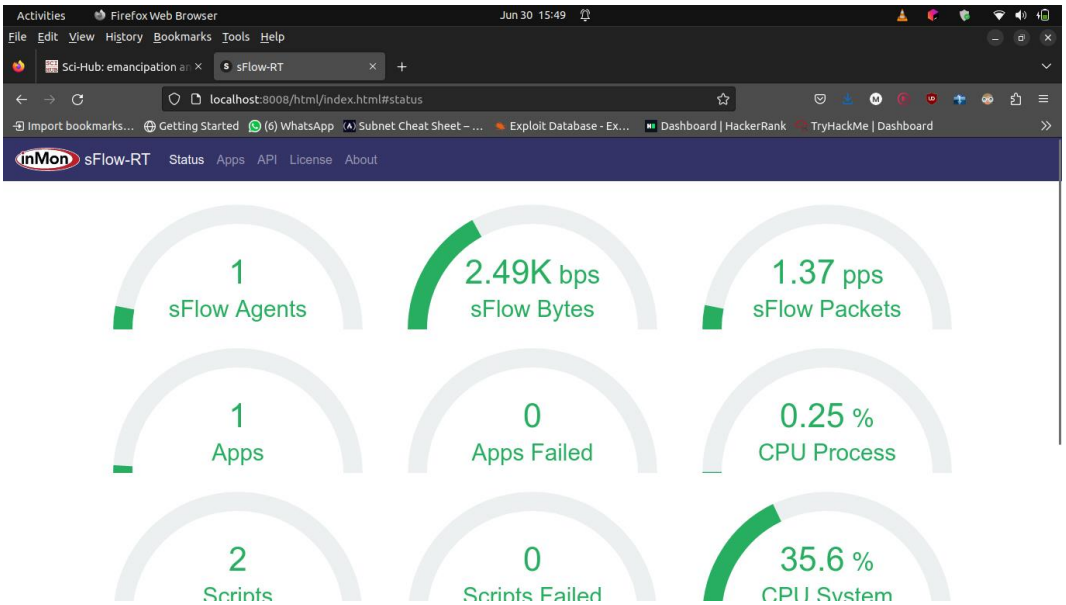


Lampiran 4: Sistem mitigasi ryu

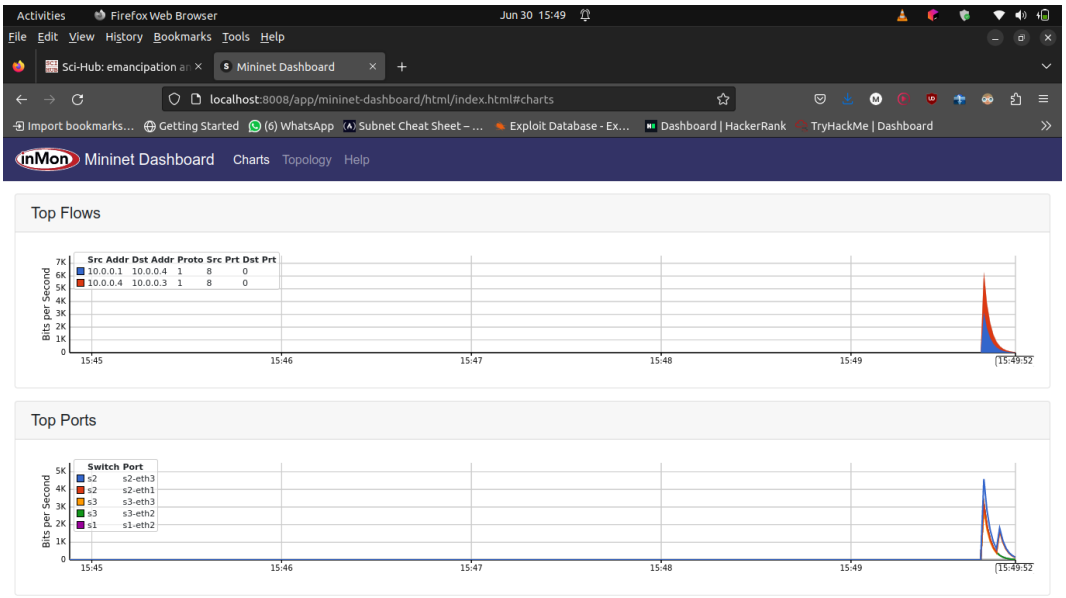


Lampiran 5: Mininet

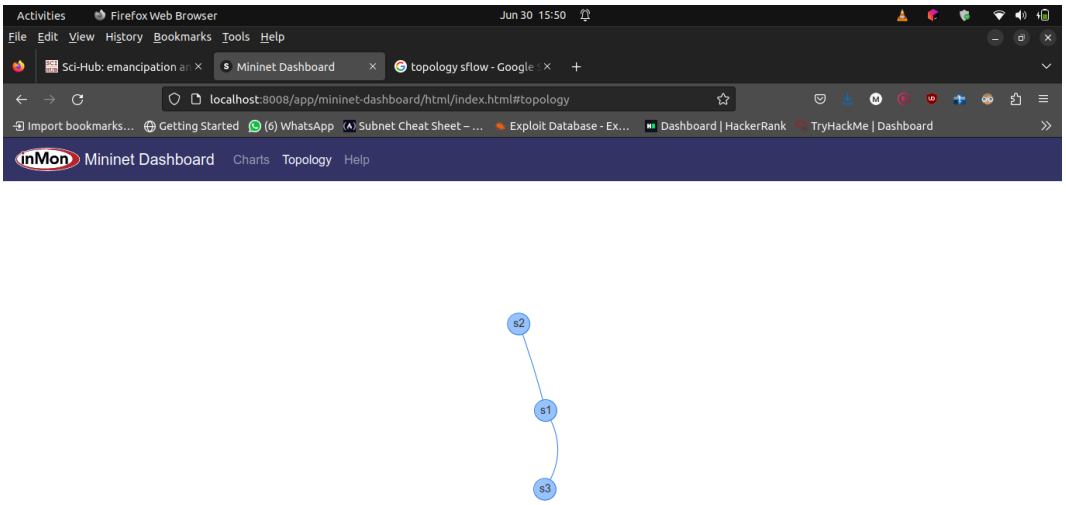
Lampiran 4 : Tampilan Dashboard sFlow



Lampiran 6: Tampilan Home



Lampiran 7: Halaman Monitor Packet



Lampiran 8: Halaman Topology

LEMBAR PERBAIKAN SKRIPSI

"PENERAPAN SOFTWARE DEFINED SECURITY

UNTUK MITIGASI DDOS"

OLEH:

ANDI MUHAMMAD GHAZY AYMAN
D121171307

Skripsi ini telah dipertahankan pada Ujian Akhir Sarjana tanggal 12 Juli 2023.

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, ST., M.IT	
Sekretaris	Dr-Eng. Ady Wahyudi Paundu, S.T., M.T.	
Anggota	Mukarramah Yusuf, B.Sc., M.Sc., Ph. D	
	Muhammad Alief Fahdal Imran Oemar, ST., M.Sc	

Persetujuan Perbaikan oleh pembimbing:

Pembimbing	Nama	Tanda Tangan
I	Dr.Eng. Muhammad Niswar, ST., M.IT	
II	Dr-Eng. Ady Wahyudi Paundu, S.T., M.T.	