

BAB 3

METODE PENELITIAN

Metodologi Penelitian berisi uraian diagram alur penelitian. Diagram alur penelitian menjelaskan mengenai tahap-tahap penelitian. Dalam penelitian ini diperlukan pula alat pendukung untuk menunjang penelitian. Selain itu penelitian ini juga melakukan perancangan topologi yang berfungsi sebagai objek pengambilan data pada proses penelitian.

3.1 ALAT YANG DIGUNAKAN

3.1.1. Perangkat Keras

Perangkat keras yang akan digunakan pada penelitian ini menggunakan satubuah laptop dengan spesifikasi sebagaimana terdapat pada tabel 3.1

Tabel 3.1 Spesifikasi Perangkat Keras

OS	Ubuntu 22.04
Processor	AMD APU A9-9400 up to 3.2 Ghz
<i>System</i> Memori (RAM)	4 GB
<i>Storage</i> (SSD)	1 TB

3.1.2. Perangkat Lunak

Perangkat lunak sebagai *tool* dan aplikasi yang digunakan pada penelitian ini dapat dilihat pada tabel 3.2

Tabel 3.2 Perangkat Lunak

No	Software	Versi	Fungsi
1	Mininet	2.3.1	Emulator SDN
2	<i>Ryu</i>	4.34	<i>Controller</i> SDN
3	<i>iperf</i>	3.6.2	Pengambilan Data Trougput
4	<i>Snort</i>	2.9.15.1	Sebagai IPS sistem

5	<i>hping3</i>	3.0.0	metode serangan
6	<i>top</i>	-	Pengambilan Data CPU dan Memori
7	Telegram	-	Monitor <i>server</i>

3.2 ALUR PENELITIAN

Penelitian ini bertujuan untuk melakukan simulasi jaringan *Software Defined Network* menggunakan *Mininet* dan *Controller Ryu* kemudian mengintegrasikan *Intrusion Prevention System* pada jaringan tersebut dan menguji sistem keamanan IPS dengan serangan DDoS. Adapun tahapan-tahapan penelitian yang akan dilakukan seperti ditunjukkan Gambar 3.1 berikut:



Gambar 3.1 Diagram Alur Penelitian

Gambar 3.1 menunjukkan diagram alur perancangan sistem dalam penelitian ini. Langkah pertama dalam penelitian yaitu melakukan studi literature beberapa penelitian terkait dengan *Software Defined Network*, DDoS, sistem keamanan IPS serta materi lain yang berhubungan dengan penelitian ini. Dengan membandingkan beberapa jurnal terkait dan melakukan perbandingan untuk menentukan judul dan juga fokus dari penelitian ini. Selain membandingkan dan menentukan fokus atau judul penelitian, tahap ini juga berfungsi untuk memahami konsep dasar dari topik tersebut.

Tahapan selanjutnya yaitu menentukan kebutuhan simulasi. Adapun yang dibutuhkan yaitu Ubuntu sebagai sistem operasi, *Ryu* sebagai *controller* serta *Firewall* dalam blokir serangan, *mininet* sebagai emulator topologi jaringan, *Snort* sebagai *Intrusion Detection System*, *hping3* sebagai metode serangan, *iperf* untuk pengambilan data QoS *throughput* dan *top* untuk pengambilan data CPU *usage* serta Memori *usage*. Selanjutnya yaitu perancangan jaringan SDN berupa rancangan topologi dan *Controller*. Pada penelitian ini menggunakan topologi *linear* yang di *custom* dan dirancang melalui *mininet*. Topologi tersusun dari empat host dimana host 3 bertindak sebagai *attacker*, host 2 dan 4 sebagai *user* dan host 1 sebagai *server* sekaligus *victim*, satu *switch* dan satu *controller*.

Setelah melakukan perancangan topologi, langkah selanjutnya perancangan sistem *security* IPS dimana pada sistem IPS menggunakan *snort* sebagai deteksi lalu lintas jaringan. *Snort* terlebih dahulu di konfigurasi sesuai dengan *rules* yang telah dibuat. *Rules* disini berfungsi sebagai monitoring jaringan dan deteksi apabila terdapat aktivasi pada *server*. Selanjutnya menyusun *bash script* yang berisi perintah blokir IP. Apabila terdeteksi serangan maka *script* blokir dijalankan dan IP *attacker* akan otomatis diblokir oleh *Controller Ryu*. Setelah perancangan sistem berhasil dibuat, tahap selanjutnya adalah menjalankan jaringan SDN dan sistem IPS untuk melihat apakah terdapat eror atau tidak. Jika terdapat *error* berupa sistem jaringan SDN yang tidak dapat dijalankan dengan *controller*, *error* pada topologi, *error* pada sistem *snort* yang belum dapat mendeteksi serangan serta *error* dalam memblokir serangan maka

perlu kembali ke penyusunan awal rancangan untuk memperbaiki *error* pada sistem tersebut. Namun jika tidak terdapat *error*, maka perancangan berhasil dan dapat dilanjutkan pada langkah pengujian serangan dan sistem yang telah dibuat.

Apabila jaringan SDN dan sistem keamanan IPS berhasil dijalankan tanpa adanya *error*, maka langkah selanjutnya adalah pengujian sistem dengan melakukan serangan DDoS menggunakan *Hping3*. Terdapat 2 tipe serangan yang digunakan yaitu *TCP Syn Flood* dan *UDP Flood*. Serangan tersebut akan dijalankan dengan variasi paket yang dikirimkan per detik dengan nilai 10, 100, 1000 dan 10.000. Dimana *server* menjadi tujuan pengujian serangan. Selanjutnya adalah pengambilan data 2 skenario berupa parameter *Qos throughput*, *CPU Usage* dan *Memori Usage*. Data yang akan diambil yaitu sebelum serangan, saat dilakukan serangan DDoS tanpa IPS dan saat dilakukan blokir serangan menggunakan sistem keamanan IPS. Proses pengambilan 2 skenario data tersebut dilakukan menggunakan *iperf* untuk pengambilan data *throughput* dan *top* untuk pengambilan data *CPU usage* serta *Memori usage*. Setelah semua data terkumpul dilanjutkan dengan melakukan analisis dan membandingkan hasil yang diperoleh dari 3 kondisi tersebut. Kemudian mengambil kesimpulan dari penelitian yang sudah dikerjakan

3.3 INTALASI KEBUTUHAN SIMULASI

Penelitian ini dilakukan secara simulasi dimana terdapat *tools* yang diperlukan dalam merancang jaringan SDN dan system keamanan IPS. Simulasi dijalankan pada sistem operasi Ubuntu yang telah diinstall secara keseluruhan pada perangkat laptop. Setelah melakukan penginstallan pada sistem operasi Ubuntu, maka selanjutnya perlu melakukan install *tools* yang akan digunakan pada penelitan ini. Diantaranya adalah:

3.3.1. Instal Ryu Beserta Service Pendukung

Ryu merupakan *controller* yang dapat mengatur ribuan jaringan secara terpusat. Selain itu, pada *ryu controller* sudah terdapat program *rest_firewall* yang dapat memblok *flow* yang tidak ingin dilewati. *Ryu controller* menggunakan bahasa

pemograman *python*, sehingga untuk menginstall *ryu* dibutuhkan *service* pendukung berbasis *python* dengan mengikuti perintah berikut:

```
#Install Ryu
sudo apt-get install python3-pip python3-dev python3-evenlet
python3-routes python3-webob python3-paramiko
sudo apt-get install git
git clone git://github.com/osrg/ryu.git
sudo pip install ryu
```

3.3.2. Install Mininet

Selanjutnya adalah melakukan install mininet, dimana mininet merupakan emulator jaringan yang nantinya berfungsi untuk membuat topologi yang diinginkan serta menjalankan jaringan SDN. Untuk menginstall mininet cukup sederhana seperti yang ditunjukkan pada perintah berikut:

```
#Install mininet
git clone https://github.com/mininet/mininet
cd mininet
git tag
git checkout -b mininet-2.3.0 2.3.0
Utile/install.sh -nfv
```

3.3.3. Install Snort

Selanjutnya melakukan instalasi *Snort*. *Snort* adalah salah satu *tools* yang digunakan dalam sistem keamanan, dimana pada penelitian ini *snort* akan dijadikan mode *Network Intrusion Detection System* (NIDS) untuk mendeteksi paket yang masuk pada *server* sesuai dengan *rules* yang telah dibuat. Cara install *snort* ditunjukkan pada perintah berikut:

```
#Install snort
sudo apt update
sudo apt-get install snort
```

3.3.4. Install Hping3

Pada penelitian ini menggunakan Hping3 sebagai pengujian serangan, dimana Hping3 salah satu *tools* yang dapat menjalankan serangan *Distributed Denial Of Service* (DDoS) yang bekerja mengirimkan paket TCP,UDP dan IP ke alamat tujuan (korban) dengan jumlah banyak secara terus menerus. Untuk menginstall *Hping3* ditunjukkan pada perintah berikut:

```
#Install Hping3
sudo apt update
sudo apt install -y hping3
```

3.3.5. Install Iperf

Iperf merupakan *tools* yang dapat mengukur kecepatan throughput seperti transfer data dan bandwidth. Untuk install *iperf* juga cukup sederhana seperti yang ditunjukkan pada perintah dibawah ini:

```
#Install Iperf
sudo apt update
sudo apt install -y iperf3
```

3.3.6. Install Top

Top merupakan salah satu *tools* yang digunakan untuk mengukur penggunaan CPU dan memori. *Software top* sudah terdapat pada sistem Linux Ubuntu sehingga tidak perlu melakukan penginstallan pada *software* ini. *Software top* dapat langsung dijalankan pada terminal Ubuntu.

3.4 RANCANGAN SISTEM

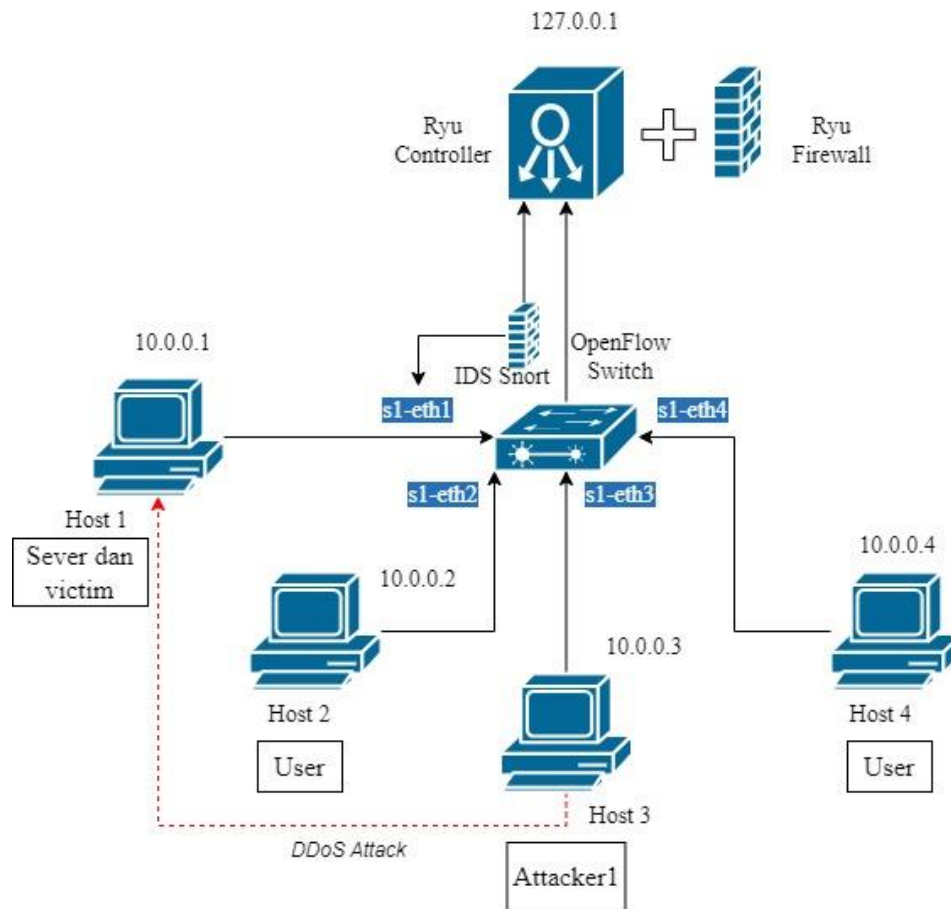
3.4.1. Perancangan Software Defined Network (SDN)

Pada perancangan *Software Defined Network* adalah merancang komponen yang akan digunakan untuk membentuk jaringan SDN. Kita ketahui bahwa dalam merancang jaringan SDN dibutuhkan sebuah *controller* sebagai pusat dan pengendali jaringan. Selain itu agar *device* dapat terhubung ke *controller* maka

dibutuhkan *OpenFlow* sebagai interface yang menjembatani *device* dengan *controller*. *OpenFlow* juga berfungsi agar setiap paket yang masuk dapat diatur secara terpusat oleh *controller*. Adapun rancangan arsitektur serta komponen yang dibutuhkan dalam merancang jaringan SDN diantaranya:

1. *Controller Ryu*
2. *OpenFlow Switch*
3. *Server*
4. *User*
5. *Attacker*

Setelah mengetahui kebutuhan komponen, maka selanjutnya merancangan topologi jaringan. Pada penelitian ini menggunakan topologi *single* yang *dicustom* pada emulator jaringan yaitu mininet. Topologi dapat dilihat pada gambar 3.2 dimana terdapat 4 host, 1 *Switch* dan 1 *Controller*. Host 1 bertindak sebagai *server* dan *victim*, kemudian host 2 dan 4 bertindak sebagai *user* sedangkan host 3 bertindak sebagai *attacker*. Garis merah putus-putus menandakan bahwa host 3 melakukan penyerangan DDoS pada *server*. Selanjutnya terdapat 1 *Switch* yang akan dijalankan sebagai *OpenFlow* agar setiap paket yang masuk dapat diatur secara terpusat oleh *Controller*. Dimana pada port *Switch* s1-eth1 terhubung pada host 1, s1-eth2 akan terhubung pada host 2, s1-eth3 akan terhubung pada host 3 dan s1-eth4 akan terhubung pada host 4.



Gambar 3.2 Rancangan Topologi jaringan

Pada port *Switch* s1-eth1 akan diintegrasikan *snort* untuk memonitoring aktivitas lalu lintas jaringan yang mengakses ke *server*. *Switch* dan *snort* akan terhubung dan dikontrol oleh *controller*. Untuk *controller* yang digunakan adalah *Ryu controller*. Setiap paket yang masuk akan diteruskan terlebih dahulu ke *Controller* untuk diperiksa apakah paket dapat teruskan atau tidak. Pada *Controller Ryu* sudah terdapat *firewall* yang bisa dijalankan secara bersamaan saat menjalankan *Ryu*. Adapun untuk pengalamanan IP Address untuk skenario pengujian adalah sebagai berikut :

Tabel 3.3 Pengalamanan IP Address

Host	Interface	IP Address
------	-----------	------------

Host 1/Server	s1-eth1	10.0.0.1/8
Host 2/User	s1-eth2	10.0.0.2/8
Host 3/Attacker	s1-eth3	10.0.0.3/8
Host 4/User	s1-eth4	10.0.0.4/8
Controller	-	127.0.0.1

Setelah mengetahui rancangan dan gambaran topologi yang akan dibuat maka langkah selanjutnya adalah membuat topologi di mininet secara *custom*. Berikut adalah perintah perancangan topologi *custom* melalui mininet.

```

from mininet.topo import Topo

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

def build( self ): "Create custom topo."

server = self.addHost( 'server', ip= '10.0.0.1' )
h2 = self.addHost('h2', ip= '10.0.0.2')
attacker1 = self.addHost( 'attacker1', ip= '10.0.0.3' )
h4 = self.addHost( 'h4', ip= '10.0.0.4' )
s1 = self.addSwitch( 's1' )

self.addLink( server, s1 )
self.addLink( h2, s1 )
self.addLink( attacker1, s1 )
self.addLink( h4, s1 )

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

```

an
bahasa pemrograman *python* dan disimpan dengan nama *topologi.py*. Topologi dibuat secara *custom* pada emulator mininet. Selain itu topologi akan dijalankan pada emulator mininet dimana *switch* akan di setting pada mode *OpenFlow* dan topologi akan di remote langsung oleh *controller*. Pada mininet sendiri sudah terdapat *controller default*. Namun pada penelitian kali ini akan menggunakan *Controller Ryu* sebagai pengontrol lalu lintas jaringan pada SDN

3.4.2. Perancangan *Intrusion Prevention System (IPS)*

Intrusion Prevention System merupakan suatu cara atau metode untuk mendeteksi dan mencegah berbagai serangan masuk ke dalam jaringan. Ada berbagai cara yang dapat digunakan dalam mengimplementasi IPS. Salah satunya adalah menerapkan *IDS Snort* atau *Suricata* untuk mendeteksi serangan dan memasang *Firewall* untuk memblokir serangan. Pada penelitian kali ini, akan dirancang *Intrusion Prevention System* menggunakan *IDS Snort* dan *Rest Firewall* yang terdapat pada *Ryu Controller*.

3.4.2.1. Perancangan IDS Snort

Pada umumnya intrusion detection system merupakan sebuah sistem yang melakukan pengawasan terhadap traffic jaringan, sedangkan *snort* merupakan perangkat lunak yang menjalankan peran dari sistem ids. *Snort* akan disimulasikan dalam jaringan SDN bertujuan untuk memonitoring trafik jaringan serta mendeteksi apabila terjadi serangan pada jaringan SDN. Sebelum menjalankan *snort* pada jaringan SDN maka terlebih dahulu melakukan konfigurasi pada *snort* dengan perintah `nano /etc/snort/snort.conf`. Konfigurasi ini bertujuan agar *snort* yang dijalankan dapat terhubung dengan jaringan SDN yang akan diintegrasikan *snort*. Namun pada konfigurasi *snort* kali ini hanya menambahkan perintah sesuai kebutuhan tanpa merubah konfigurasi default dari *snort*. Berikut adalah gambar konfigurasi *snort* yang akan dijalankan pada *snort*.

```
#ipvar HOME_NET any  
ipvar HOME_NET 10.0.0.1/8
```

Pada perintah diatas melakukan konfigurasi dengan menambah alamat IP *server* 10.0.0.1/8 sebagai IP yang akan dilindungi

```
#For more information, see snort manual, configuring snort - output modules  
output alert_csv:/home/wenny/ips/alerts.csv timestamp,msg,src,dst
```

Pada perintah diatas melakukan penambahan perintah Output Alert_csv berupa waktu, pesan, *source* dan *destination*. Agar log yang dicatat oleh *snort* disimpan dalam bentuk file csv yang mana nantinya file tersebut akan dipanggil untuk menampilkan monitoring deteksi serangan serta sebagai data untuk memblokir serangan yang masuk.

```
#site spesific rules
include $RULE_PATH/local.rules
```

Kemudian pada perintah diatas menambahkan include `$RULE_PATH/local.rules` sebagai *specific rules*, dimana bertujuan agar *rules* yang dibaca *snort* berfokus pada *local.rules*. Setelah proses konfigurasi *snort* berhasil dan tidak terjadi *error*, maka selanjutnya dapat menambahkan *rules* seperti yang ditunjukkan pada perintah dibawah. *Snort* akan bereaksi apabila ada *rules* yang cocok dengan paket yang masuk berupa notifikasi *alert*, yang mana *snort* akan mencatat terlebih dahulu log *alert* yang masuk berupa waktu, pesan, *source*, *destination* dan menyimpan dalam bentuk file csv pada *directory /home/wenny/ips/* yang diberi nama *Alerts.csv*. File *Alerts.csv* akan dipanggil pada terminal untuk melihat *alert* dari serangan.

```
#Aktivasi Ping
alert icmp any any <> any any (msg : "Aktivasi Normal"; sid:1000001;rev:0;)

#Tcp Syn Flood
alert tcp any any -> $HOME_NET any (flags: S; msg:"Possible DDOS Attack
Type : SYNflood"; flow:stateless; sid:10002; detection_filter:track by_dst,
count 10, seconds 10;)

#UDP Flood
alert udp any any -> $HOME_NET any (threshold: type threshold, track
by_src, count 10, seconds 10; msg:"Possible DDOS Attack Type : UDP
Flood";sid:10000007;rev:2;)
```

Terdapat 3 *rules* yang diintegrasikan pada *snort*. *Rules* pertama saat aktivasi normal, dimana *rules* tersebut memberikan aturan apabila terdapat paket ICMP dari IP mana saja menuju IP *server*, maka akan muncul *alerts* “Aktivasi Normal”. Untuk *rules*

kedua adalah deteksi serangan TCP SYN *Flood* dengan aturan *rules* apabila terdapat host dan port dari manapun menuju *server* dengan melakukan pengiriman lebih dari 10 paket TCP selama periode sampling 1 detik menuju IP *destination* yang dilindungi dengan port tujuan 80 dan tipe *flags* S maka akan muncul *alert DDoS detection Type: TCP SYN Flood*. Kemudian untuk *rules* terakhir adalah deteksi serangan DDoS dengan tipe UDP. Apabila terdapat host dan port dari manapun menuju *server* dengan melakukan pengiriman lebih dari 10 paket UDP selama periode sampling 10 detik menuju *server* akan muncul *alert DDoS detection Type: UDP Flood*.

Tabel 3.4 Penjelasan Perintah *Rules*

Alert	action yang diberikan <i>snort</i> ketika <i>rules</i> sesuai
icmp/tcp/udp	protokol yang akan dideteksi
any	<i>snort</i> akan melihat sumber IP dari mana saja
any	<i>snort</i> akan melihat semua port dari mana saja
<>	menunjukkan arah antara dua address
->	menunjukkan arah dari <i>source</i> menuju <i>destination</i>
\$HOME_NET	Alamat IP tujuan, sesuai dengan konfigurasi <i>snort</i>
any	<i>snort</i> melihat tujuan ip dari mana saja
any	<i>snort</i> melihat port <i>destination</i> dari mana saja
80	port yang akan dituju yaitu port 80
msg	pesan yang akan ditampilkan pada <i>alert</i>
sid:1000001	ID Rule yang dimulai dari sid:1000001
rev:0	Nomor revisi untuk memudahkan dalam aturan
flags: S	TCP mengirimkan paket dengan tipe flag SYN
detection_filter:track by_dst	<i>snort</i> melacak alamat IP tujuan untuk dideteksi
seconds 10	periode sampling
count 10	jika selama periode sampling <i>snort</i> mendeteksi 10 atau lebih, maka akan muncul <i>alert</i> (nilai count dapat diganti)

Selain *alert snort* yang dipanggil pada terminal, penulis juga menambahkan *alert* yang terhubung pada telegram agar memudahkan admin dalam memonitoring *server*. Untuk menghubungkan *alert* pada telegram, maka dibuatlah perintah notifikasi *alert* yang ditunjukkan perintah dibawah ini:

```
#!/bin/bash

initCount=0

logs=/home/wenny/ips/alerts.csv

#File
msg_caption=/tmp/telegram_msg_caption.txt

#Chat ID dan bot token Telegram
chat_id="-1002005801598"
token="6831773016:AAHEg6OO46X4qaoiAartSFbG90COjNPCKVs"

# kirim
function sendAlert
{
    curl -s -F chat_id=$chat_id -F text="$caption"
https://api.telegram.org/bot\$token/sendMessage #> /dev/null 2&>1
}
}
```

Kemudian dilanjutkan dengan perintah seperti di bawah ini:

```
while true
do
    lastCount=$(wc -c $logs | awk '{print $1}')
    if(($lastCount) > $initCount);
    then
        msg=$(tail -n 2 $logs) #GetLastLineLog
        echo -e "Halo Admin\nTerjadi aktivasi pada Server, segera
periksa!!!\n\nServer Time : $(date +"%d %b %Y %T")\n\n"$msg >
$msg_caption #set Caption / Pesan
        caption=$(<$msg_caption) #set Caption
        sendAlert #Panggil Fungsi di function
        echo "Alert Terkirim"
        initCount=$lastCount
        rm -f $msg_caption
        sleep 1
    fi
    sleep 2 #delay if Not Indication
done
```

Perintah diatas berfungsi untuk menampilkan notifikasi *alert* yang terhubung dengan telegram. Bot terlebih dahulu dibuat agar mendapatkan token serta *chat ID* yang berfungsi mengirimkan notifikasi pada telegram. File log csv yang telah dicatat dan disimpan *snort* pada *alerts.csv* akan dipanggil pada *script* dan ditampilkan dalam bentuk pesan yang dikirimkan oleh bot ke Grup Monitoring Server yang dibuat oleh penulis sebelumnya.

3.4.2.2. Perancangan *Rest Firewall*

Pada dasarnya *Firewall* merupakan sebuah sistem yang berfungsi untuk mengizinkan lalu lintas jaringan yang dianggap aman dan mencegah serta memblok lalu lintas yang dianggap berbahaya. Keuntungan menggunakan *firewall* adalah jaringan dapat terlindung dari serangan yang tidak diinginkan, mengontrol dan membatasi akses jaringan yang boleh dilewati. Jaringan SDN sendiri sangat rentan terhadap serangan. Sehingga perlu dirancang *firewall* untuk melindungi jaringan serta memblok serangan yang masuk. Pada penelitian ini akan menggunakan *rest_firewall* dari *Ryu Controller*. *Rest_firewall* pada *Ryu* akan dijalankan secara bersamaan dengan *Controller Ryu*. Agar *rest_firewall* pada *Ryu* dapat berjalan baik, maka dibuatlah perintah untuk melakukan bloking serangan terlebih dahulu.

```
logfile="/home/wenny/ips/alerts.csv"

tail -n 1 -f $logfile | while read line; do
newTime=`echo $line | cut -f 1 -d ","`
x_anc=`echo $line | cut -f 2 -d ","`
src=`echo $line | cut -f 3 -d ","`
dst=`echo $line | cut -f 4 -d ","`
printf "\n"

blockTime=$(date +"%m/%d %H:%M:%S")

echo "IP Source : "$src
echo "IP Destination : "$dst
echo "Type ancaman : "$x_anc
echo "Waktu masuk : "$newTime
echo "Waktu blokir : "$blockTime

./blokir.sh $src $dst $x_anc &
sleep 5
done
```

Pada perintah diatas merupakan *script* perintah program berbasis *Bash*. Pada dasarnya Script *block.sh* berfungsi melakukan pemeriksaan serangan melalui file log csv yang telah dibuat *snort* sebelumnya. File tersebut akan dipanggil pada *script* ini, guna memblokir serangan yang masuk. Perintah *tail -n 1 -f* akan membaca file log yang masuk secara *realtime*. Berupa waktu masuk serangan atau *newTime*, tipe ancaman, *source*, *destination* dan waktu blokir. *Script block.sh* akan membaca waktu terakhir paket masuk sehingga *script* hanya akan dijalankan apabila pada *alerts.csv* mendeteksi ada serangan yang masuk.

```
#!/bin/bash

src=$1
dst=$2
x_anc=$4

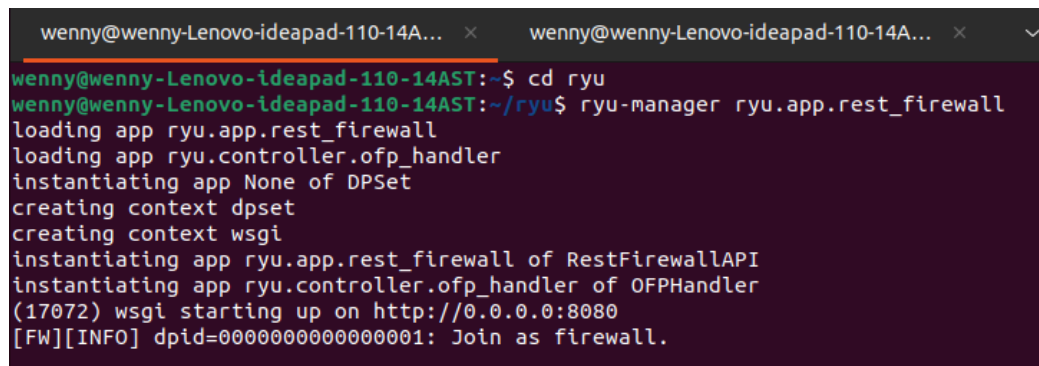
curl -s -X POST -d
 '{"nw_src":"$src","nw_dst":"$dst","actions":"DENY","priority": "2"}'
 http://localhost:8080/firewall/rules/0000000000000001

curl -s -X POST -d
 '{"nw_src":"$dst","nw_dst":"$src","actions":"DENY","priority": "2"}'
 http://localhost:8080/firewall/rules/0000000000000001
```

Pada perintah di atas merupakan *script* *blokir.sh* yang berfungsi untuk memblokir serangan. *Script* *blokir.sh* menjalankan perintah *firewall* yang telah disediakan *ryu* untuk membatasi atau memblokir ip yang dianggap berbahaya. “`{ "nw_src": "$src", "nw_dst": "$dst", "actions": "deny", "priority": "2" }`’ `http://localhost:8080/firewall/rules/0000000000000001`” merupakan perintah blokir yang terdapat pada *firewall controller ryu* yang dijalankan pada *script* *blokir.sh*. *Script* ini akan dijalankan bersamaan dengan *block.sh* dimana *script* ini hanya akan dipanggil oleh *block.sh* untuk memblokir serangan yang masuk.

3.5 PENGUJIAN SISTEM

Pada tahap ini, akan dilakukan pengujian jaringan *software defined network* dan keamanan *intrusion prevention system* yang telah dirancang sebelumnya. Langkah pertama yang harus dijalankan adalah *Controller Ryu*. Untuk menjalankan *Controller Ryu* terlebih dahulu masuk ke directory *ryu* dengan perintah `cd ryu`. Kemudian

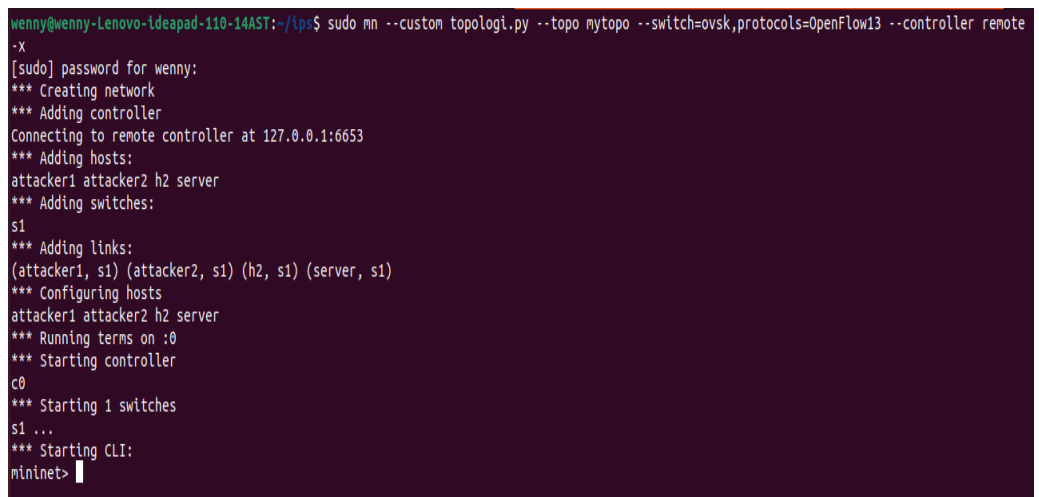


```
wenny@wenny-Lenovo-ideapad-110-14A... x wenny@wenny-Lenovo-ideapad-110-14A... x v
wenny@wenny-Lenovo-ideapad-110-14AST:~$ cd ryu
wenny@wenny-Lenovo-ideapad-110-14AST:~/ryu$ ryu-manager ryu.app.rest_firewall
loading app ryu.app.rest_firewall
loading app ryu.controller.ofp_handler
instantiating app None of DPSet
creating context dpset
creating context wsgi
instantiating app ryu.app.rest_firewall of RestFirewallAPI
instantiating app ryu.controller.ofp_handler of OFPHandler
(17072) wsgi starting up on http://0.0.0.0:8080
[FW][INFO] dpid=0000000000000001: Join as firewall.
```

jalankan *controller ryu* bersamaan dengan *rest_firewall* dengan perintah `ryu-manager ryu.app.rest_firewall`, maka akan muncul tampilan seperti pada gambar 3.3

Gambar 3.3 Running Ryu

Gambar 3.3 menunjukkan bahwa *Controller Ryu* telah berhasil dijalankan bersamaan dengan *rest_firewall*. Secara otomatis *Controller Ryu* akan mengaktifkan perintah *firewall* sehingga semua paket yang masuk akan langsung diblokir oleh *controller ryu*. Selanjutnya jalankan topologi yang telah dibuat menggunakan mininet dengan cara masuk ke directory *ips* menggunakan perintah `cd ips`. Kemudian jalankan perintah `sudo mn --custom topologi.py --topo mytopo --switch=ovsk,protocols=OpenFlow13 --controller remote -x`. Maka akan muncul



```
wenny@wenny-Lenovo-ideapad-110-14AST:~/ips$ sudo mn --custom topologi.py --topo mytopo --switch=ovsk,protocols=OpenFlow13 --controller remote -x
[sudo] password for wenny:
*** Creating network
*** Adding controller
Connecting to remote controller at 127.0.0.1:6653
*** Adding hosts:
attacker1 attacker2 h2 server
*** Adding switches:
s1
*** Adding links:
(attacker1, s1) (attacker2, s1) (h2, s1) (server, s1)
*** Configuring hosts
attacker1 attacker2 h2 server
*** Running terms on :0
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```


tampilan pada Gambar 3.4

Gambar 3.4 Running Topologi pada Mininet

Gambar 3.4 menampilkan bahwa topologi telah berhasil dijalankan pada mininet. Perintah *sudo mn* berfungsi menjalankan topologi pada mininet, *--custom topologi.py* merupakan perintah topologi yang *dicustom* sendiri dengan nama *topologi.py*. Kemudian untuk memanggil topologi yang telah *dicustom* sebelumnya menggunakan perintah *-topo mytopo*. Untuk perintah *-switch=ovsk,protocols=OpenFlow* menjalankan *switch* pada mode *ovsk* dan protokol *OpenFlow*. Sedangkan perintah *-controller remote* dijanakan agar host dan *switch* dapat terhubung dan kontrol oleh *Ryu*. Perintah *-x* berfungsi untuk memunculkan external terminal dari semua host, *switch* dan *controller*. Setelah dijalankan topologi pada mininet maka akan muncul beberapa external terminal yang nantinya aktivitas serangan akan dilakukan pada terminal tersebut. Kemudian mencoba melakukan pingall pada mininet, maka akan muncul tampilan seperti dibawah ini:

```
mininet> pingall
*** Ping: testing ping reachability
attacker1 -> X X X
attacker2 -> X X X
h2 -> X X X
server -> X X X
*** Results: 100% dropped (0/12 received)
mininet> █
```

Gambar 3.5 Hasil Pingall

Pada gambar 3.5 terlihat bahwa hasil ping menunjukkan *reachability* dikarenakan semua paket telah di drop oleh *Controller Ryu*. Agar host dapat saling terhubung satu sama lain maka dibuatlah perintah *script allowflow.sh* yang mana pada *script* tersebut berisikan perintah *rules allow* dari *controller ryu* sehingga host dapat mengirim paket dan saling terhubung satu sama lain:

```
GNU nano 6.2 allowFlow.sh
curl -X PUT http://localhost:8080/firewall/module/enable/0000000000000001
curl -X POST -d '{"nw_src": "10.0.0.1","nw_dst": "10.0.0.2"}' http://localhost:8080/firewall/rules/0000000000000001
curl -X POST -d '{"nw_src": "10.0.0.1","nw_dst": "10.0.0.3"}' http://localhost:8080/firewall/rules/0000000000000001
curl -X POST -d '{"nw_src": "10.0.0.1","nw_dst": "10.0.0.4"}' http://localhost:8080/firewall/rules/0000000000000001
curl -X POST -d '{"nw_src": "10.0.0.2","nw_dst": "10.0.0.3"}' http://localhost:8080/firewall/rules/0000000000000001
curl -X POST -d '{"nw_src": "10.0.0.2","nw_dst": "10.0.0.4"}' http://localhost:8080/firewall/rules/0000000000000001
curl -X POST -d '{"nw_src": "10.0.0.3","nw_dst": "10.0.0.4"}' http://localhost:8080/firewall/rules/0000000000000001
curl -X POST -d '{"nw_src": "10.0.0.2","nw_dst": "10.0.0.1"}' http://localhost:8080/firewall/rules/0000000000000001
curl -X POST -d '{"nw_src": "10.0.0.3","nw_dst": "10.0.0.1"}' http://localhost:8080/firewall/rules/0000000000000001
curl -X POST -d '{"nw_src": "10.0.0.4","nw_dst": "10.0.0.1"}' http://localhost:8080/firewall/rules/0000000000000001
curl -X POST -d '{"nw_src": "10.0.0.3","nw_dst": "10.0.0.2"}' http://localhost:8080/firewall/rules/0000000000000001
curl -X POST -d '{"nw_src": "10.0.0.4","nw_dst": "10.0.0.2"}' http://localhost:8080/firewall/rules/0000000000000001
curl -X POST -d '{"nw_src": "10.0.0.4","nw_dst": "10.0.0.3"}' http://localhost:8080/firewall/rules/0000000000000001
```

Gambar 3.6 Perintah *Allowflow.Sh*

```
wenny@wenny-Lenovo-ideapad-110-14AST:~/ips$ ./allowFlow.sh
[{"switch_id": "0000000000000001", "command_result": {"result": "success", "details": "firewall running."}}][{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=1"}]}][{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=2"}]}][{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=3"}]}][{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=4"}]}][{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=5"}]}][{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=6"}]}][{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=7"}]}][{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=8"}]}][{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=9"}]}][{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=10"}]}][{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=11"}]}][{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=12"}]}]wenny@wenny-Lenovo-ideapad-110-14AST:~/ips$
```

Gambar 3.7 Running *AllowFlow.Sh*

Kemudian jalankan perintah *allowflow.sh* pada directory ips, maka akan muncul gambar 3.7, dari hasil perintah tersebut setiap host telah diizinkan. Selain itu, pada *Controller Ryu* akan muncul notifikasi bahwa semua host telah beri izin oleh *Controller Ryu* sesuai *rules* yang telah dibuat. Ditunjukkan gambar 3.8

```
loading app ryu.app.rest_firewall
loading app ryu.controller.ofp_handler
instantiating app None of DPSet
creating context dpset
creating context wsgi
instantiating app ryu.app.rest_firewall of RestFirewallAPI
instantiating app ryu.controller.ofp_handler of OFPHandler
(17072) wsgi starting up on http://0.0.0.0:8080
[FW][INFO] dpid=0000000000000001: Join as firewall.
(17072) accepted ('127.0.0.1', 50026)
127.0.0.1 - - [07/May/2024 15:34:51] "PUT /firewall/module/enable/0000000000000001 HTTP/1.1" 200 217 0.004274
(17072) accepted ('127.0.0.1', 50034)
127.0.0.1 - - [07/May/2024 15:34:51] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 225 0.001204
(17072) accepted ('127.0.0.1', 50046)
127.0.0.1 - - [07/May/2024 15:34:51] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 225 0.001054
(17072) accepted ('127.0.0.1', 50056)
127.0.0.1 - - [07/May/2024 15:34:51] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 225 0.001169
(17072) accepted ('127.0.0.1', 50068)
127.0.0.1 - - [07/May/2024 15:34:51] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 225 0.001050
(17072) accepted ('127.0.0.1', 50078)
127.0.0.1 - - [07/May/2024 15:34:51] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 225 0.001044
(17072) accepted ('127.0.0.1', 50080)
127.0.0.1 - - [07/May/2024 15:34:51] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 225 0.001098
(17072) accepted ('127.0.0.1', 50084)
127.0.0.1 - - [07/May/2024 15:34:51] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 225 0.001139
(17072) accepted ('127.0.0.1', 50088)
127.0.0.1 - - [07/May/2024 15:34:51] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 225 0.002515
(17072) accepted ('127.0.0.1', 50090)
127.0.0.1 - - [07/May/2024 15:34:51] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 225 0.003073
(17072) accepted ('127.0.0.1', 50100)
127.0.0.1 - - [07/May/2024 15:34:51] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 226 0.001055
(17072) accepted ('127.0.0.1', 50114)
127.0.0.1 - - [07/May/2024 15:34:51] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 226 0.001148
(17072) accepted ('127.0.0.1', 50124)
127.0.0.1 - - [07/May/2024 15:34:51] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 226 0.001640
```

Gambar 3.8 *Controller Ryu* memberi izin Host

```
mininet> pingall
*** Ping: testing ping reachability
attacker1 -> attacker2 h2 server
attacker2 -> attacker1 h2 server
h2 -> attacker1 attacker2 server
server -> attacker1 attacker2 h2
*** Results: 0% dropped (12/12 received)
mininet> █
```

Gambar 3.9 Hasil pingall setelah ditambahkan *allowFlow.sh*

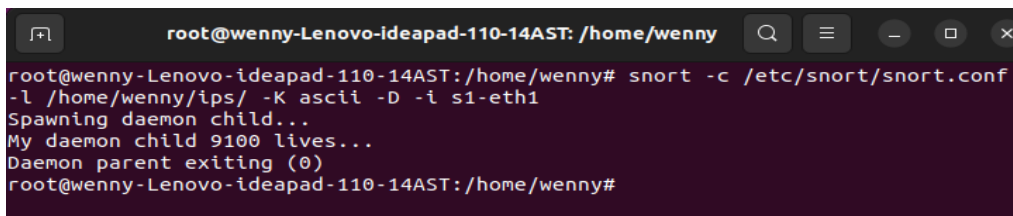
Selanjutnya lakukan pingall kembali pada mininet, maka setiap host sudah dapat terhubung satu sama lain. Agar Host 1 terintegrasi *server*, maka pada terminal host 1 jalankan perintah *python3 -m http.server 80* seperti Gambar 3.10 dimana host 1 telah berhasil dijalankan sebagai *server* http yang berjalan pada port 80.



```
root@wenny-Lenovo-ideapad-110-14AST:/home/wenny/ips# python3 -m http.server 80
Servicing HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
█
```

Gambar 3.10 Menjalankan *server* pada Host 1

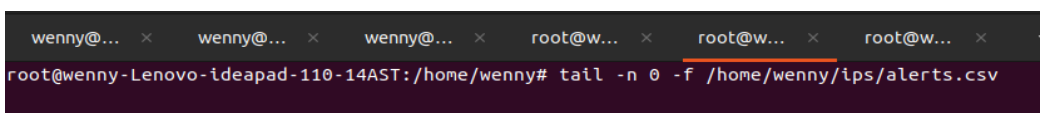
Kemudian jalankan *snort* dengan masuk terlebih dahulu ke superadmin menggunakan perintah *sudo su* lalu ketikkan perintah *snort -c /etc/snort/snort.conf -l /home/wenny/ips/alert -K ascii -D -i s1-eth1* seperti pada Gambar 3.11.



```
root@wenny-Lenovo-ideapad-110-14AST:/home/wenny# snort -c /etc/snort/snort.conf
-l /home/wenny/ips/ -K ascii -D -i s1-eth1
Spawning daemon child...
My daemon child 9100 lives...
Daemon parent exiting (0)
root@wenny-Lenovo-ideapad-110-14AST:/home/wenny#
```

Gambar 3.11 Running *snort*

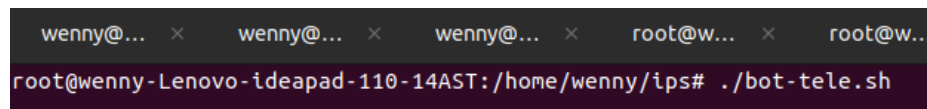
Pada gambar 3.11 terlihat bahwa *snort* telah berhasil dijalankan. *Snort* akan menjalankan perintah sesuai dengan konfigurasi dan *rules* yang telah dibuat sebelumnya. Pada perintah ini menjalankan *snort* dalam mode *ascii* agar local *rules* bisa dibaca dengan mudah. Kemudian *snort* juga dijalankan pada *s1-eth1* yang mana



```
root@wenny-Lenovo-ideapad-110-14AST:/home/wenny# tail -n 0 -f /home/wenny/ips/alerts.csv
```

interface tersebut terhubung ke *server*. *Snort* akan memeriksa semua paket yang masuk melewati *server*, apabila sesuai dengan *rules* yang telah dibuat pada */etc/snort/rules/local.rules*. Maka akan menampilkan *alert* yang mana *alert* tersebut dicatat oleh *snort* dan terlebih dahulu disimpan pada folder */home/wenny/ips/alerts.csv*. Untuk memanggil *alert* dapat menjalankan perintah pada *tail -f /home/wenny/ips/alerts.csv* seperti Gambar 3.12.

Gambar 3. 12. Menampilkan Alert pada folder *alerts.csv*



```
wenny@... x wenny@... x wenny@... x root@w... x root@w...
root@wenny-Lenovo-ideapad-110-14AST:/home/wenny/ips# ./bot-tele.sh
```

Gambar

3.13 Menampilkan *alert* pada telegram

Selain itu, juga ditambahkan *alert* yang terhubung pada telegram dengan menjalankan perintah *./bot-tele.sh* pada directory *ips*. Menampilkan *alert* pada telegram bertujuan memudahkan admin dalam memonitoring aktivitas *server*. Sehingga apabila terjadi serangan pada *server* maka dapat langsung dimitigasi. Agar memastikan sistem dapat berjalan baik, maka host 2,3 dan 4 melakukan ping IP pada *server* terlebih dahulu seperti yang ditunjukkan pada Gambar 3.14, setiap host telah berhasil melakukan ping IP pada *server*.

```
root@wenny-Lenovo-ideapad-110-14AST:/home/wenny/ips# ping -c 1 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data:
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.850 ms

--- 10.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.850/0.850/0.850/0.000 ms
root@wenny-Lenovo-ideapad-110-14AST:/home/wenny/ips#

root@wenny-Lenovo-ideapad-110-14AST:/home/wenny/ips# ping -c 1 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data:
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.693 ms

--- 10.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.693/0.693/0.693/0.000 ms
root@wenny-Lenovo-ideapad-110-14AST:/home/wenny/ips#

root@wenny-Lenovo-ideapad-110-14AST:/home/wenny/ips# ping -c 1 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data:
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.656 ms

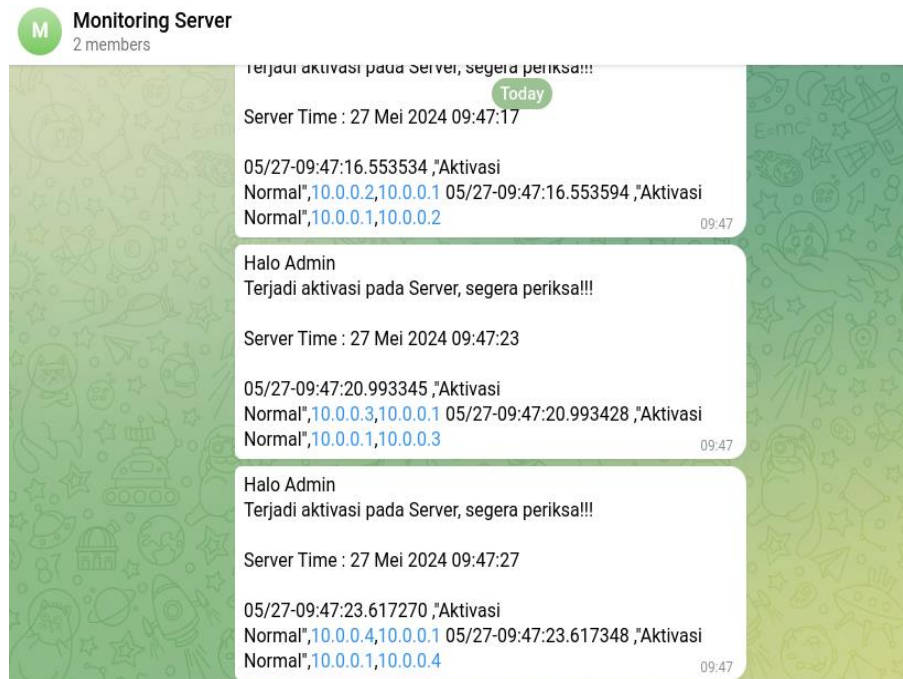
--- 10.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.656/0.656/0.656/0.000 ms
root@wenny-Lenovo-ideapad-110-14AST:/home/wenny/ips#
```

Gambar 3.14 Ping host pada server

Selanjutnya pada gambar 3.14 menunjukkan reaksi *snort* terhadap ping yang telah dilakukan. Terlihat file log csv yang dipanggil sebelumnya menampilkan *alert* “Aktivasi Normal” dari masing-masing host.

```
wenny@... x wenny@... x wenny@... x root@w... x root@w... x root@w... x
root@wenny-Lenovo-ideapad-110-14AST:/home/wenny# tail -n 0 -f /home/wenny/ips/alerts.csv
05/27-09:47:16.553534 ,"Aktivasi Normal",10.0.0.2,10.0.0.1
05/27-09:47:16.553594 ,"Aktivasi Normal",10.0.0.1,10.0.0.2
05/27-09:47:20.993345 ,"Aktivasi Normal",10.0.0.3,10.0.0.1
05/27-09:47:20.993428 ,"Aktivasi Normal",10.0.0.1,10.0.0.3
05/27-09:47:23.617270 ,"Aktivasi Normal",10.0.0.4,10.0.0.1
05/27-09:47:23.617348 ,"Aktivasi Normal",10.0.0.1,10.0.0.4
```

Gambar 3.15 Menampilkan *alert* dari Folder Alerts.csv



Gambar

3.16 Alert pada telegram

Kemudian pada telegram juga akan muncul notifikasi *alert* seperti pada Gambar 3.16. Dapat dilihat waktu yang ditampilkan pada telegram dan *alerts.csv* sama dan *realtime* menandakan bahwa sistem jaringan SDN telah berhasil dijalankan.

3.6 PENGUJIAN SERANGAN

Pada tahap ini akan dilakukan pengujian serangan pada jaringan *Software Defined Network*. Dalam pengujian serangan ini yang bertindak sebagai korban adalah Host 1 yang telah diintegrasikan sebagai *server*. Sedangkan host 3 akan bertindak sebagai penyerang. Pengujian serangan menggunakan DDoS dengan tipe serangan TCP SYN *Flood* dan UDP *Flood*.

3.6.1. Pengujian Serangan TCP SYN *Flood*

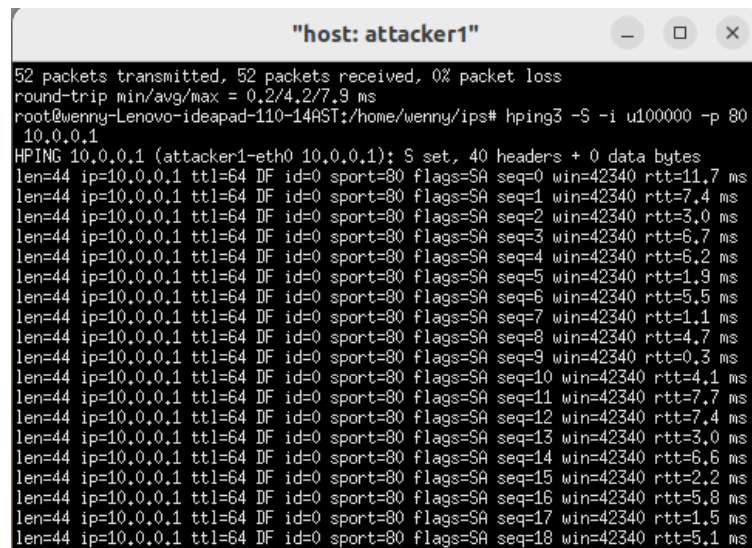
Setelah berhasil menginstall Hping3 pada sistem, pengujian serangan sudah dapat dilakukan. Serangan pertama yang akan dilakukan adalah DDoS TCP SYN *Flood*. Host 3 terlebih dahulu melakukan ping pada IP *server* untuk mengetahui apakah host tersebut dapat mengakses *server*. Terlihat pada gambar 3.17 Menunjukkan

host 3 dapat mengakses *server*. Selanjutnya menjalankan perintah `hping3 -S -I u100000 -p 80 10.0.0.1`. Seperti yang ditunjukkan pada Gambar 3.18.



```
root@wenny-Lenovo-ideapad-110-14AST:/home/wenny/ips# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data:
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.584 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.130 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.161 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.124 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.145 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.131 ms
64 bytes from 10.0.0.1: icmp_seq=7 ttl=64 time=0.121 ms
64 bytes from 10.0.0.1: icmp_seq=8 ttl=64 time=0.116 ms
64 bytes from 10.0.0.1: icmp_seq=9 ttl=64 time=0.151 ms
64 bytes from 10.0.0.1: icmp_seq=10 ttl=64 time=0.110 ms
^C
--- 10.0.0.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 920ms
rtt min/avg/max/mdev = 0.110/0.177/0.584/0.136 ms
root@wenny-Lenovo-ideapad-110-14AST:/home/wenny/ips#
```

Gambar 3.17 Hasil ping host 3 ke *server*



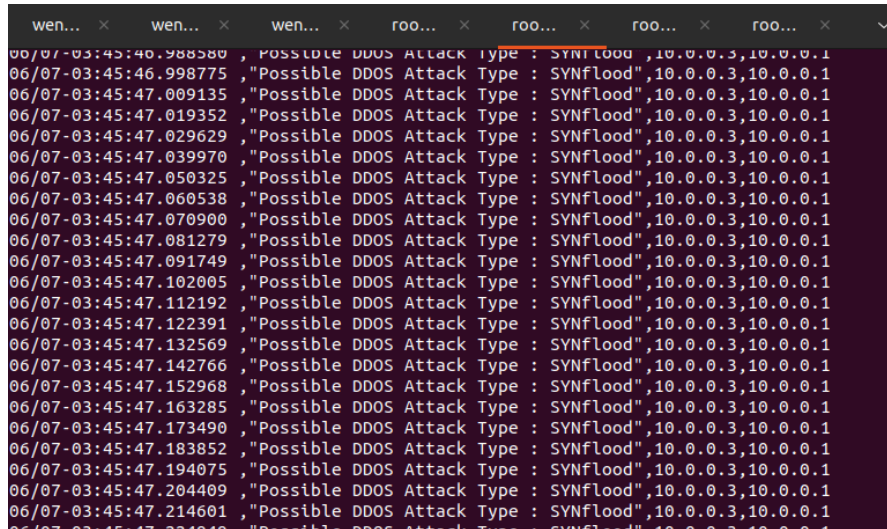
```
52 packets transmitted, 52 packets received, 0% packet loss
round-trip min/avg/max = 0.2/4.2/7.9 ms
root@wenny-Lenovo-ideapad-110-14AST:/home/wenny/ips# hping3 -S -i u100000 -p 80
10.0.0.1
HPING 10.0.0.1 (attacker1-eth0 10.0.0.1): S set, 40 headers + 0 data bytes
len=44 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=SA seq=0 win=42340 rtt=11.7 ms
len=44 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=SA seq=1 win=42340 rtt=7.4 ms
len=44 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=SA seq=2 win=42340 rtt=3.0 ms
len=44 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=SA seq=3 win=42340 rtt=6.7 ms
len=44 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=SA seq=4 win=42340 rtt=6.2 ms
len=44 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=SA seq=5 win=42340 rtt=1.9 ms
len=44 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=SA seq=6 win=42340 rtt=5.5 ms
len=44 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=SA seq=7 win=42340 rtt=1.1 ms
len=44 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=SA seq=8 win=42340 rtt=4.7 ms
len=44 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=SA seq=9 win=42340 rtt=0.3 ms
len=44 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=SA seq=10 win=42340 rtt=4.1 ms
len=44 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=SA seq=11 win=42340 rtt=7.7 ms
len=44 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=SA seq=12 win=42340 rtt=7.4 ms
len=44 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=SA seq=13 win=42340 rtt=3.0 ms
len=44 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=SA seq=14 win=42340 rtt=6.6 ms
len=44 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=SA seq=15 win=42340 rtt=2.2 ms
len=44 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=SA seq=16 win=42340 rtt=5.8 ms
len=44 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=SA seq=17 win=42340 rtt=1.5 ms
len=44 ip=10.0.0.1 ttl=64 DF id=0 sport=80 flags=SA seq=18 win=42340 rtt=5.1 ms
```

Gambar 3.18 Perintah serangan TCP SYN Flood

Arti dari perintah tersebut:

1. `hping3`: Tools yang menjalankan perintah pengiriman paket
2. `-S`: Menandakan paket yang dikirim berasal dari protokol TCP SYN
3. `-I`: Interval Paket

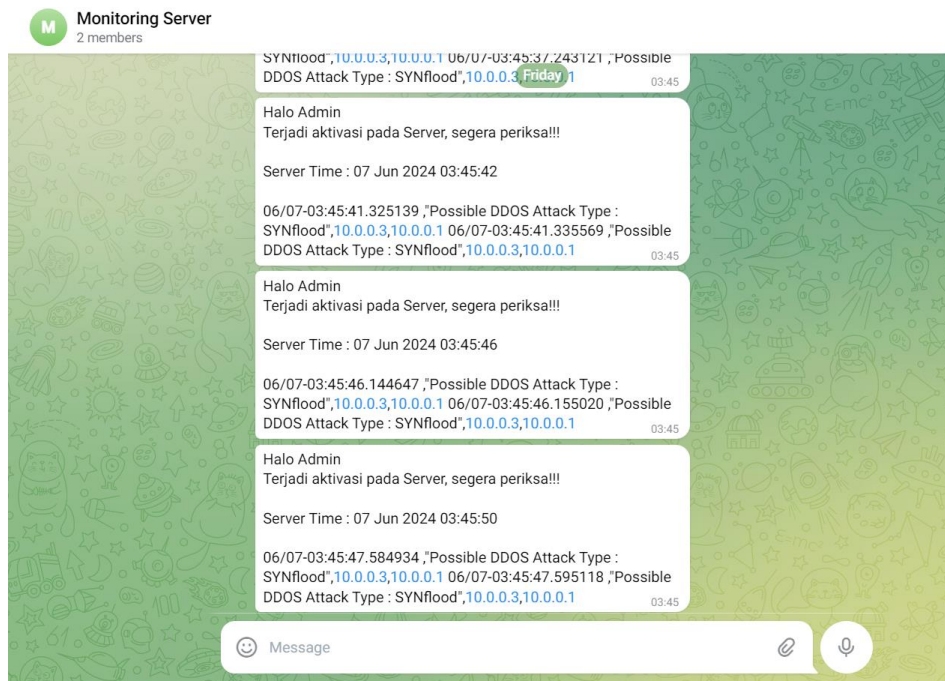
4. `-u100000`: interval paket yang dikirim sebanyak 10 paket perdetik
5. `-p`: Tujuan port yang akan diserang
6. `10.0.0.1`: IP tujuan yang diserang (IP *server*)



```
wen... x wen... x wen... x roo... x roo... x roo... x roo... x
06/07-03:45:46.988580 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
06/07-03:45:46.998775 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
06/07-03:45:47.009135 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
06/07-03:45:47.019352 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
06/07-03:45:47.029629 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
06/07-03:45:47.039970 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
06/07-03:45:47.050325 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
06/07-03:45:47.060538 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
06/07-03:45:47.070900 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
06/07-03:45:47.081279 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
06/07-03:45:47.091749 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
06/07-03:45:47.102005 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
06/07-03:45:47.112192 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
06/07-03:45:47.122391 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
06/07-03:45:47.132569 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
06/07-03:45:47.142766 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
06/07-03:45:47.152968 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
06/07-03:45:47.163285 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
06/07-03:45:47.173490 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
06/07-03:45:47.183852 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
06/07-03:45:47.194075 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
06/07-03:45:47.204409 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
06/07-03:45:47.214601 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
06/07-03:45:47.224800 , "Possible DDoS Attack Type : SYNflood", 10.0.0.3, 10.0.0.1
```

Gambar 3.19 Hasil deteksi *snort* melalui folder Alerts.csv

Setelah berhasil menjalankan perintah serangan TCP SYN Flood. Terlihat pada log *alerts.csv* menampilkan *alert* yang ditunjukkan pada Gambar 3.19. Di mana *alert* yang ditunjukkan terjadi serangan pada IP *server* yaitu IP 10.0.0.1 dengan ditandai pesan DDoS “ *Possible DDoS Attack Type: SYN Flood*” yang dilakukan oleh IP 10.0.0.3 serta diikuti dengan waktu masuknya paket ke *server*. Selain itu juga dapat dilihat pada bot telegram yang ditunjukkan pada Gambar 3.20



Gambar 3.20 Hasil notifikasi *snort* pada Telegram

Pada Gambar 3.20 menunjukkan notifikasi serangan masuk ke telegram melalui pesan yang dikirimkan oleh bot. Dari log *alerts.csv* yang dipanggil pada terminal serta notifikasi yang dikirimkan ke telegram, menandakan *rules* yang telah dibuat untuk mendeteksi serangan DDoS TCP SYN *Flood* berhasil dijalankan *snort*.

3.6.2. Pengujian Serangan UDP *Flood*

Selanjutnya adalah pengujian serangan UDP *Flood*, dimana alur pengujian ini sama dengan pengujian serangan TCP SYN *Flood* Sebelumnya. Langkah pertama host 3 melakukan ping IP terlebih dahulu ke *server*. Apabila paket berhasil dikirimkan ke *server*, menandakan bahwa host 3 dapat mengakses *server* seperti yang ditunjukkan pada gambar 3.21. Setelah berhasil melakukan ping IP pada *server*, maka selanjutnya menjalankan perintah `hping3 10.0.0.1 -udp -p 80 -I u100000` seperti yang ditunjukkan pada Gambar 3.22

```
"host: attacker1"
root@wenny-Lenovo-ideapad-110-14AST:/home/wenny/ips# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data:
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.584 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.130 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.161 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.124 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.145 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.131 ms
64 bytes from 10.0.0.1: icmp_seq=7 ttl=64 time=0.121 ms
64 bytes from 10.0.0.1: icmp_seq=8 ttl=64 time=0.116 ms
64 bytes from 10.0.0.1: icmp_seq=9 ttl=64 time=0.151 ms
64 bytes from 10.0.0.1: icmp_seq=10 ttl=64 time=0.110 ms
```

Gambar 3.21 Hasil ping Host 3 ke server

```
"host: attacker1"
--- 10.0.0.1 hping statistic ---
110 packets transmitted, 16 packets received, 86% packet loss
round-trip min/avg/max = 0,9/4,5/8,0 ms
root@wenny-Lenovo-ideapad-110-14AST:/home/wenny/ips# hping3 10.0.0.1 --udp -p
80 -i u100000
HPING 10.0.0.1 (attacker1-eth0 10.0.0.1): udp mode set, 28 headers + 0 data byte
s
ICMP Port Unreachable from ip=10.0.0.1 name=UNKNOWN
status=0 port=2089 seq=0
ICMP Port Unreachable from ip=10.0.0.1 name=UNKNOWN
status=0 port=2090 seq=1
ICMP Port Unreachable from ip=10.0.0.1 name=UNKNOWN
status=0 port=2091 seq=2
ICMP Port Unreachable from ip=10.0.0.1 name=UNKNOWN
status=0 port=2092 seq=3
ICMP Port Unreachable from ip=10.0.0.1 name=UNKNOWN
status=0 port=2093 seq=4
```

Gambar 3.22 serangan UDP Dilancarkan

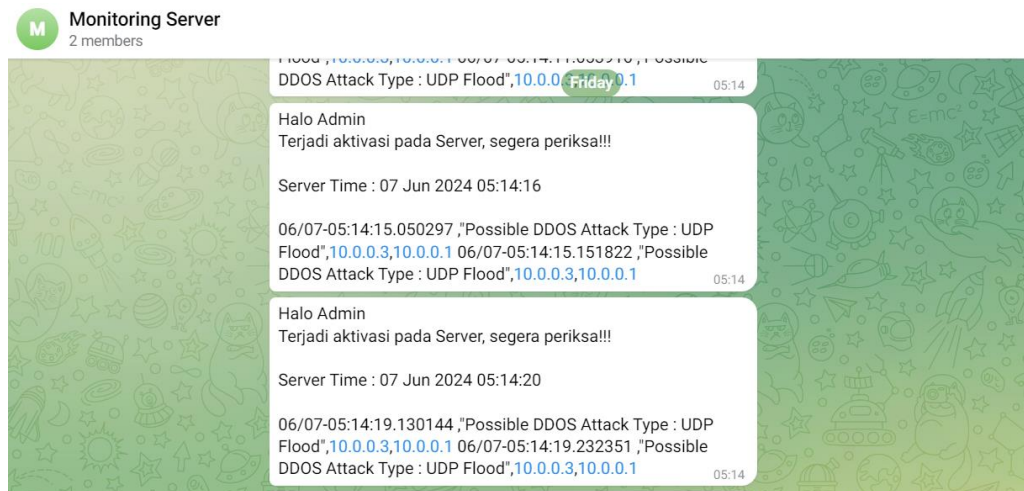
Arti perintah tersebut:

1. Hping3: Tools yang menjalankan perintah mengirimkan paket
2. 10.0.0.1: IP tujuan yang di serang (IP server)
3. -udp: Tipe paket UDP
4. -p: Tujuan port yang akan diserang yaitu port 80
5. -I: Interval Paket
6. -u100000: interval paket yang dikirim sebanyak 10 paket perdetik

```
06/07-05:14:17.901282 , "Possible DDOS Attack Type : UDP Flood", 10.0.0.3, 10.0.0.1
06/07-05:14:18.003392 , "Possible DDOS Attack Type : UDP Flood", 10.0.0.3, 10.0.0.1
06/07-05:14:18.105526 , "Possible DDOS Attack Type : UDP Flood", 10.0.0.3, 10.0.0.1
06/07-05:14:18.207781 , "Possible DDOS Attack Type : UDP Flood", 10.0.0.3, 10.0.0.1
06/07-05:14:18.311183 , "Possible DDOS Attack Type : UDP Flood", 10.0.0.3, 10.0.0.1
06/07-05:14:18.413184 , "Possible DDOS Attack Type : UDP Flood", 10.0.0.3, 10.0.0.1
06/07-05:14:18.515834 , "Possible DDOS Attack Type : UDP Flood", 10.0.0.3, 10.0.0.1
06/07-05:14:18.618052 , "Possible DDOS Attack Type : UDP Flood", 10.0.0.3, 10.0.0.1
06/07-05:14:18.720095 , "Possible DDOS Attack Type : UDP Flood", 10.0.0.3, 10.0.0.1
06/07-05:14:18.822319 , "Possible DDOS Attack Type : UDP Flood", 10.0.0.3, 10.0.0.1
```

Gambar 3.23 Alert.csv saat serangan UDP

Setelah berhasil dijalankan serangan *UDP Flood* maka akan muncul *alert* seperti pada Gambar 3.23 yang menunjukkan *alert* terjadi serangan UDP pada IP server yaitu IP 10.0.0.1 dengan ditandai pesan DDoS “Possible DDoS Attack Type: UDP Flood” yang dilakukan oleh IP 10.0.0.3 serta diikuti dengan waktu masuknya paket ke server. Selain itu juga dapat dilihat pada bot telegram yang ditunjukkan pada Gambar 3.24



Gambar 3.24 Notikasi serangan UDP pada Telegram

Pada Gambar 3.24 menunjukkan notifikasi serangan masuk ke telegram melalui pesan yang dikirimkan oleh bot. Dari log *alerts.csv* yang dipanggil pada terminal serta notifikasi yang dikirimkan ke telegram, menandakan *rules* yang telah dibuat untuk mendeteksi serangan DDoS *UDP Flood* berhasil dijalankan *snort*.

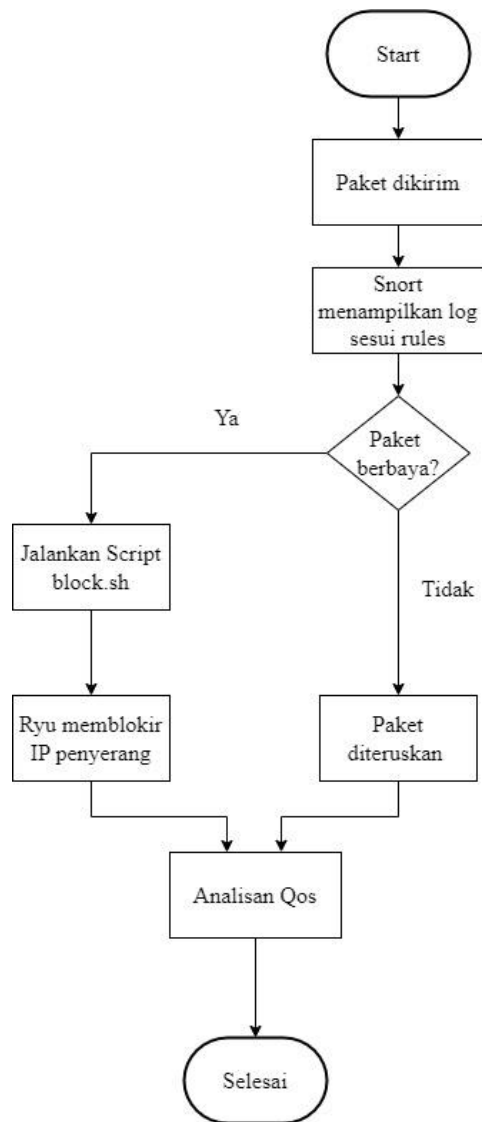
1.7 PENGUJIAN SISTEM KEMAMAN IPS

Pada Penelitian ini akan dilakukan pengujian sistem kewanaman IPS. *Intrusion Prevention System* (IPS) akan disimulasikan pada jaringan *Software Defined Network* (SDN) dengan menggunakan IDS *Snort* untuk mendeteksi serangan yang masuk ke

server. *Snort* akan dijalankan pada *Switch* interface *si-eth1* yang terhubung ke Host 1 atau *server*. Apabila terdapat paket yang masuk sesuai dengan *rules* yang dibuat, maka *snort* akan mencatat log tersebut dan menyimpan log dalam bentuk file csv dimana file tersebut akan dipanggil pada terminal dan telegram dalam bentuk *alert* dan notifikasi. Data yang akan dicatat *snort* berupa waktu, pesan, *source* dan *destination*.

Apabila terdapat paket data berbahaya yang masuk menuju *server*, pemblokiran tidak akan terjadi secara langsung oleh *snort*, dikarenakan *snort* hanya mencatat log dan mendeteksi yang kemudian memberikan notifikasi kepada admin apabila terjadi serangan. Agar pemblokiran pada serangan dapat terintegrasi maka dijalankan *ryu rest_firewall* untuk memblokir serangan yang masuk. Terdapat 2 *script* yang dijalankan pada *Ryu rest_firewall* yaitu *block.sh* dan *blokir.sh*.

Pada dasarnya *block.sh* hanya melakukan pemeriksaan serangan melalui file log csv yang telah dibuat *snort* sebelumnya. File *block.sh* akan membaca waktu terakhir paket masuk dan dikirimkan ke file *blokir.sh* untuk dijalankan pemblokiran pada IP. Pada *script blokir.sh* berisikan perintah *firewall* yang telah disediakan *Ryu* untuk membatasi atau memblokir IP yang dianggap berbahaya. Script *blokir.sh* berada didalam *script block.sh* dan dijalankan bersamaan dengan *block.sh*. Script *firewall* ini hanya akan dijalankan apabila pada *alerts.csv* mendeteksi serangan masuk.



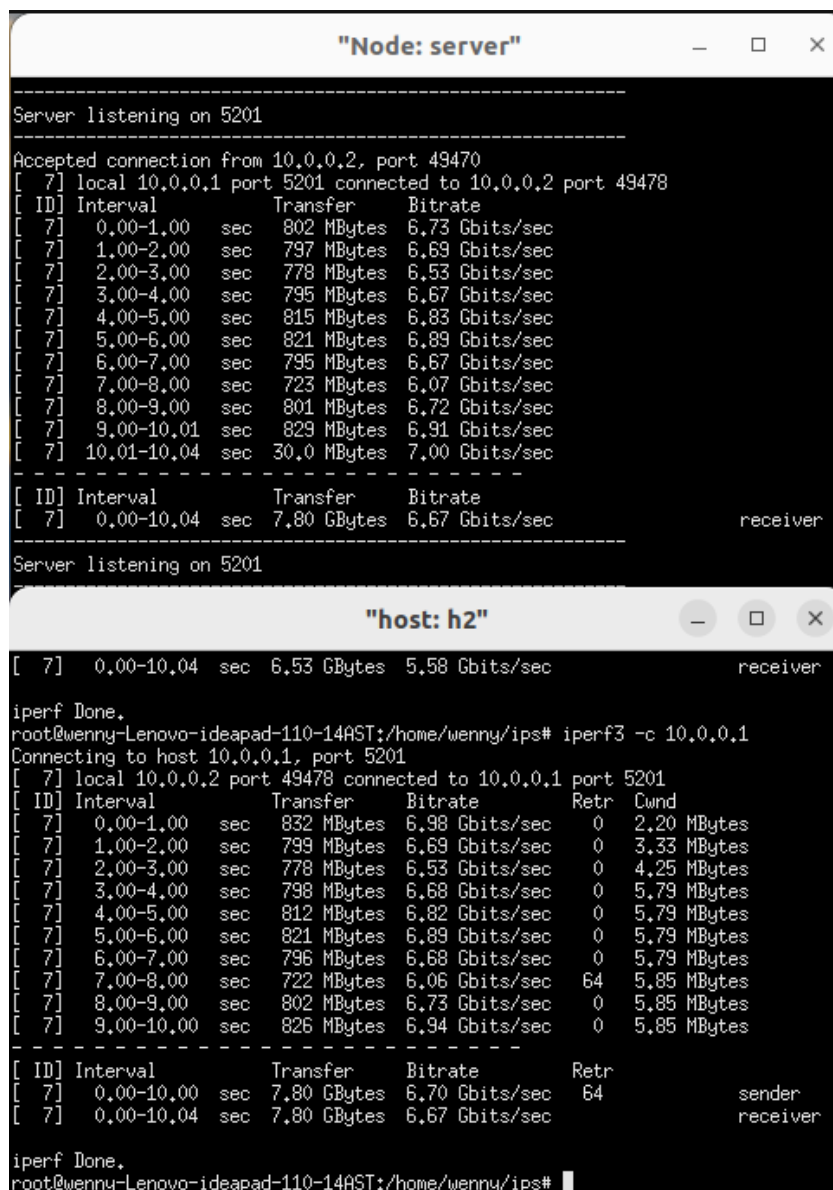
Gambar 3.25 Flowchart Sistem keamanan IPS

Gambar 3.25 merupakan alur sistem kemanan IPS yang telah dirancang. Saat adanya paket yang masuk menuju *server snort* akan menampilkan log sesuai *rules*. Apabila paket tersebut tidak berbahaya , maka paket akan diteruskan ke *server* dan dilakukan analisa QoS. Namun apabila paket yang masuk dianggap berbahaya, maka admin dapat menjalankan *script /block.sh* yang telah dibuat sebelumnya. Saat Script berhasil dijalankan, *Controller Ryu* akan melakukan pemblokiran pada IP penyerang dan dilakukan analisa data dari hasil yang didapatkan. Tahap ini penulis akan melakukan pengujian sistem kemanan IPS dengan melakukan serangan pada *server*.

Untuk melihat apakah sistem berjalan baik atau tidak. Setelah semua komponen yang dibutuhkan pada jaringan SDN berhasil dijalankan. Maka dilanjutkan dengan menjalankan *snort* pada *interface switch s1-eth1*, file *Alerts.csv* dan *script bot-tele.sh*. Agar setiap paket yang masuk ke *server* dapat dideteksi ditampilkan pada file *Alerts.csv* serta Telegram. Pada pengujian ini akan menggunakan serangan 1000 paket TCP SYN *Flood* ke *server*. Terlebih dahulu host 2 akan melakukan test bandwidth pada *server*. Pengecekan ini untuk mengetahui hasil data transfer dan bandwidth yang didapatkan sebelum dilakukan serangan. Gambar 3.26 menunjukan hasil *iperf* ke

server dalam keadaan normal dengan nilai bandwidth sebesar 6,67 Gbps dan transfer sebesar 7,80 GBps

Gambar 3.26 Hasil Bandwidth dan Transfer server saat normal



The image shows two terminal windows. The top window, titled "Node: server", displays the output of an iperf3 test. It shows the server listening on port 5201, accepting a connection from 10.0.0.2, and then reporting performance metrics for 10-second intervals and a total summary. The bottom window, titled "host: h2", shows the client side of the test, including the iperf3 command execution and its corresponding performance metrics.

```
-----
Server listening on 5201
-----
Accepted connection from 10.0.0.2, port 49470
[ 7] local 10.0.0.1 port 5201 connected to 10.0.0.2 port 49478
[ ID] Interval      Transfer      Bitrate
[ 7] 0.00-1.00    sec  802 MBytes  6.73 Gbits/sec
[ 7] 1.00-2.00    sec  797 MBytes  6.69 Gbits/sec
[ 7] 2.00-3.00    sec  778 MBytes  6.53 Gbits/sec
[ 7] 3.00-4.00    sec  795 MBytes  6.67 Gbits/sec
[ 7] 4.00-5.00    sec  815 MBytes  6.83 Gbits/sec
[ 7] 5.00-6.00    sec  821 MBytes  6.89 Gbits/sec
[ 7] 6.00-7.00    sec  795 MBytes  6.67 Gbits/sec
[ 7] 7.00-8.00    sec  723 MBytes  6.07 Gbits/sec
[ 7] 8.00-9.00    sec  801 MBytes  6.72 Gbits/sec
[ 7] 9.00-10.01   sec  829 MBytes  6.91 Gbits/sec
[ 7] 10.01-10.04  sec  30.0 MBytes  7.00 Gbits/sec
-----
[ ID] Interval      Transfer      Bitrate
[ 7] 0.00-10.04   sec  7.80 GBytes  6.67 Gbits/sec
-----
Server listening on 5201

-----
"host: h2"
-----
[ 7] 0.00-10.04   sec  6.53 GBytes  5.58 Gbits/sec
-----
iperf Done.
root@wenny-Lenovo-ideapad-110-14AST:/home/wenny/ips# iperf3 -c 10.0.0.1
Connecting to host 10.0.0.1, port 5201
[ 7] local 10.0.0.2 port 49478 connected to 10.0.0.1 port 5201
[ ID] Interval      Transfer      Bitrate      Retr  Cwnd
[ 7] 0.00-1.00    sec  832 MBytes  6.98 Gbits/sec  0    2.20 MBytes
[ 7] 1.00-2.00    sec  799 MBytes  6.69 Gbits/sec  0    3.33 MBytes
[ 7] 2.00-3.00    sec  778 MBytes  6.53 Gbits/sec  0    4.25 MBytes
[ 7] 3.00-4.00    sec  798 MBytes  6.68 Gbits/sec  0    5.79 MBytes
[ 7] 4.00-5.00    sec  812 MBytes  6.82 Gbits/sec  0    5.79 MBytes
[ 7] 5.00-6.00    sec  821 MBytes  6.89 Gbits/sec  0    5.79 MBytes
[ 7] 6.00-7.00    sec  796 MBytes  6.68 Gbits/sec  0    5.79 MBytes
[ 7] 7.00-8.00    sec  722 MBytes  6.06 Gbits/sec  64   5.85 MBytes
[ 7] 8.00-9.00    sec  802 MBytes  6.73 Gbits/sec  0    5.85 MBytes
[ 7] 9.00-10.00   sec  826 MBytes  6.94 Gbits/sec  0    5.85 MBytes
-----
[ ID] Interval      Transfer      Bitrate      Retr
[ 7] 0.00-10.00   sec  7.80 GBytes  6.70 Gbits/sec  64
[ 7] 0.00-10.04   sec  7.80 GBytes  6.67 Gbits/sec
-----
iperf Done.
root@wenny-Lenovo-ideapad-110-14AST:/home/wenny/ips#
```

```

"host: attacker1"
52 packets transmitted, 52 packets received, 0% packet loss
round-trip min/avg/max = 0,2/4,2/7,9 ms
root@wenny-Lenovo-ideapad-110-14AST:/home/wenny/ips# hping3 -S -i u100000 -p 80
10.0.0.1
HPING 10.0.0.1 (attacker1-eth0 10.0.0.1): S set, 40 headers + 0 data bytes

```

Gambar 3.27 Attacker melakukan penyerangan ke server

Selanjutnya host 3 akan bertindak sebagai attacker dengan mengirimkan serangan 1000 paket TCP Syin Flood seperti yang ditunjukkan pada gambar 3.27 dimana serangan TCP Syin Flood berhasil dijalankan. Selanjutnya dapat jalan kan *iperf* kembali dari host 2 menuju *server* dilihat pada Gambar 3.28 nilai bandwidth menurun menjadi 3,84 Gbps dan transfer menjadi 4,49 GBps. Dapat dilihat serangan 1000 TCP SYN Flood membanjiri *server* dengan permintaan yang mengakibatkan beban *server* menjadi berat dan menghabiskan sumber daya *server*.

```

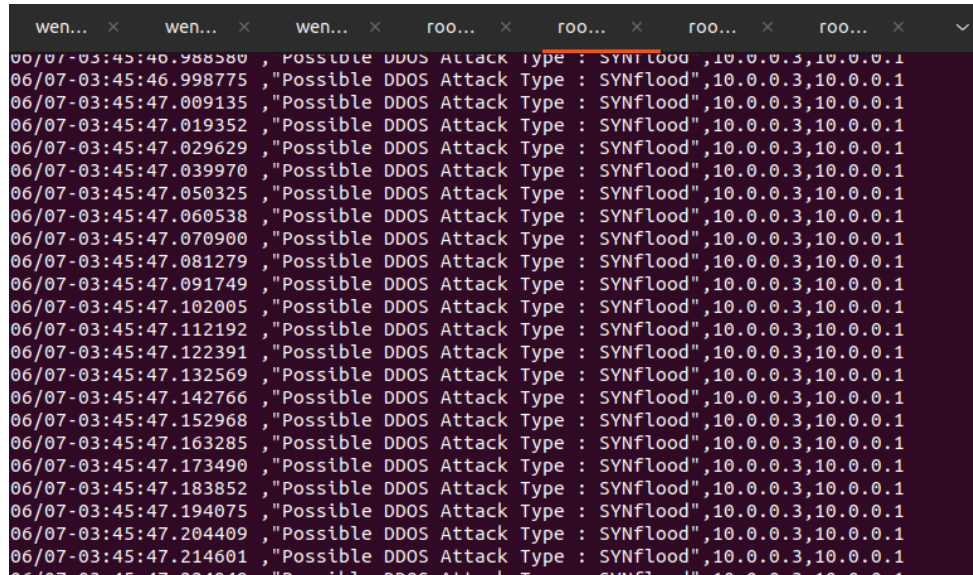
"Node: server"
[ 7] 0,00-10,04 sec 5,70 GBytes 4,87 Gbits/sec receiver
-----
Server listening on 5201
-----
Accepted connection from 10.0.0.2, port 34878
[ 7] local 10.0.0.1 port 5201 connected to 10.0.0.2 port 34884
[ ID] Interval      Transfer      Bitrate
[ 7] 0,00-1,00 sec 544 MBytes 4,56 Gbits/sec
[ 7] 1,00-2,00 sec 454 MBytes 3,81 Gbits/sec
[ 7] 2,00-3,00 sec 478 MBytes 4,01 Gbits/sec
[ 7] 3,00-4,00 sec 401 MBytes 3,35 Gbits/sec
[ 7] 4,00-5,00 sec 379 MBytes 3,19 Gbits/sec
[ 7] 5,00-6,00 sec 454 MBytes 3,81 Gbits/sec
[ 7] 6,00-7,00 sec 714 MBytes 5,99 Gbits/sec
[ 7] 7,00-8,00 sec 229 MBytes 1,92 Gbits/sec
[ 7] 8,00-9,00 sec 532 MBytes 4,46 Gbits/sec
[ 7] 9,00-10,00 sec 386 MBytes 3,24 Gbits/sec
[ 7] 10,00-10,04 sec 31,4 MBytes 6,10 Gbits/sec
-----
[ ID] Interval      Transfer      Bitrate
[ 7] 0,00-10,04 sec 4,49 GBytes 3,84 Gbits/sec receiver
-----

"host: h2"
iperf Done.
root@wenny-Lenovo-ideapad-110-14AST:/home/wenny/ips# iperf3 -c 10.0.0.1
Connecting to host 10.0.0.1, port 5201
[ 7] local 10.0.0.2 port 34884 connected to 10.0.0.1 port 5201
[ ID] Interval      Transfer      Bitrate      Retr  Cwnd
[ 7] 0,00-1,00 sec 568 MBytes 4,76 Gbits/sec 45 2,11 MBytes
[ 7] 1,00-2,00 sec 445 MBytes 3,73 Gbits/sec 305 2,22 MBytes
[ 7] 2,00-3,00 sec 481 MBytes 4,04 Gbits/sec 0 2,23 MBytes
[ 7] 3,00-4,00 sec 408 MBytes 3,42 Gbits/sec 0 2,25 MBytes
[ 7] 4,00-5,00 sec 368 MBytes 3,08 Gbits/sec 0 2,26 MBytes
[ 7] 5,00-6,00 sec 461 MBytes 3,88 Gbits/sec 0 2,29 MBytes
[ 7] 6,00-7,00 sec 725 MBytes 6,09 Gbits/sec 0 2,34 MBytes
[ 7] 7,00-8,00 sec 209 MBytes 1,75 Gbits/sec 0 2,34 MBytes
[ 7] 8,00-9,00 sec 535 MBytes 4,49 Gbits/sec 0 2,36 MBytes
[ 7] 9,00-10,00 sec 402 MBytes 3,38 Gbits/sec 0 2,38 MBytes
-----
[ ID] Interval      Transfer      Bitrate      Retr
[ 7] 0,00-10,00 sec 4,49 GBytes 3,86 Gbits/sec 350
[ 7] 0,00-10,04 sec 4,49 GBytes 3,84 Gbits/sec
sender
receiver
iperf Done.
root@wenny-Lenovo-ideapad-110-14AST:/home/wenny/ips# iperf3 -c 10.0.0.1

```

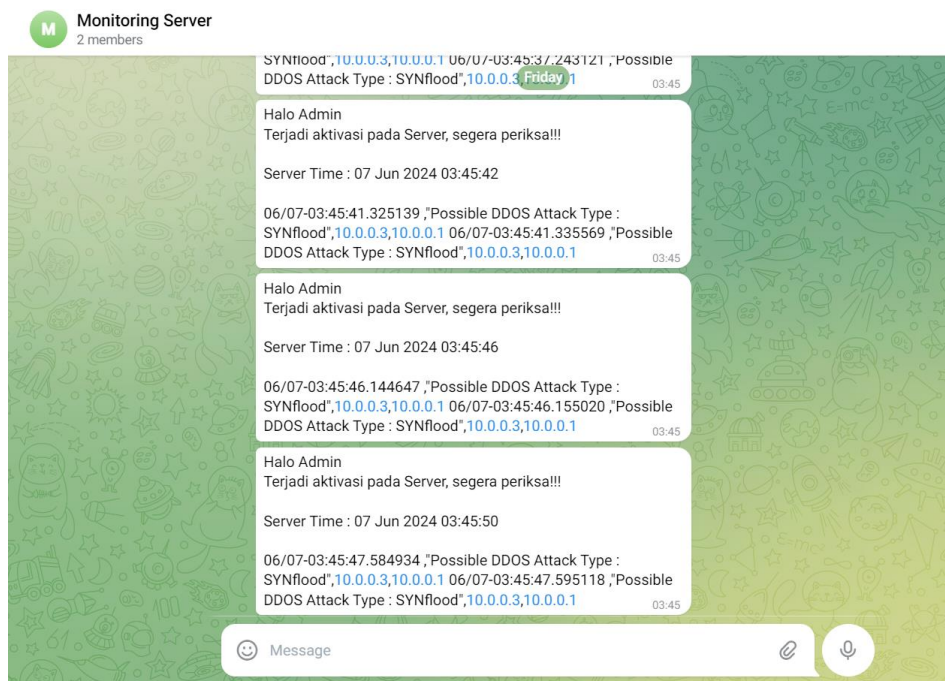

Gambar 3.28 Hasil ping IP host 2 ke server saat diserang

Saat serangan berlangsung, dapat dilihat log *alerts.csv* yang dipanggil menampilkan pesan *alert Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1* serta notifikasi pesan yang dikirimkan oleh *bot-tele* ke Telegram. Seperti yang ditunjukkan pada Gambar. 3.29 dan 3.30



```
wen... x wen... x wen... x roo... x roo... x roo... x roo... x
06/07-03:45:46.988580 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:45:46.998775 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:45:47.009135 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:45:47.019352 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:45:47.029629 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:45:47.039970 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:45:47.050325 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:45:47.060538 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:45:47.070900 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:45:47.081279 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:45:47.091749 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:45:47.102005 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:45:47.112192 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:45:47.122391 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:45:47.132569 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:45:47.142766 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:45:47.152968 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:45:47.163285 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:45:47.173490 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:45:47.183852 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:45:47.194075 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:45:47.204409 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:45:47.214601 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:45:47.224810 , Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
```

Gambar 3.29 Alert pada log *alerts CSV*



Gambar 3.30 Notifikasi *snort* pada Telegram

```

root@wenny-Lenovo-ideapad-110-14AST:/home/wenny/ips# ./block.sh
IP Source          : 10.0.0.3
IP Destination     : 10.0.0.1
Type ancaman       : "Possible DDOS Attack Type : SYNflood"
Waktu masuk        : 06/07-03:30:14.686320
Waktu blokir       : 06/07 03:30:14
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=13"}]} [{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=14"}]}
IP Source          : 10.0.0.3
IP Destination     : 10.0.0.1
Type ancaman       : "Possible DDOS Attack Type : SYNflood"
Waktu masuk        : 06/07-03:30:14.786509
Waktu blokir       : 06/07 03:30:19
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=15"}]} [{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=16"}]}

```

Gambar 3.31 Menjalankan *script* block.sh untuk blokir serangan

Selanjutnya, setelah mengetahui adanya serangan admin dapat menjalankan

```

[FW][INFO] dpid=0000000000000001: Blocked packet = ethernet(dst='4a:4f:e1:10:4d:b3', ethertype=2048, src='4a:64:c7:14:5e:f0'), ipv4(csum=12658, dst='10.0.0.1', flags=2, header_length=5, identification=62771, offset=0, option=None, proto=1, src='10.0.0.3', tos=0, total_length=84, ttl=64, version=4), icmp(code=0, csum=55776, data=echo(data=b'\x7f\x00\x00\x00:\x05\x01\x00\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#%&\'()*+,-./01234567', id=26282, seq=195), type=8)
[FW][INFO] dpid=0000000000000001: Blocked packet = ethernet(dst='4a:4f:e1:10:4d:b3', ethertype=2048, src='4a:64:c7:14:5e:f0'), ipv4(csum=12574, dst='10.0.0.1', flags=2, header_length=5, identification=62855, offset=0, option=None, proto=1, src='10.0.0.3', tos=0, total_length=84, ttl=64, version=4), icmp(code=0, csum=47489, data=echo(data=b'\x7f\x00\x00\x00:\x05\x01\x00\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#%&\'()*+,-./01234567', id=26282, seq=196), type=8)
[FW][INFO] dpid=0000000000000001: Blocked packet = ethernet(dst='4a:4f:e1:10:4d:b3', ethertype=2048, src='4a:64:c7:14:5e:f0'), ipv4(csum=12342, dst='10.0.0.1', flags=2, header_length=5, identification=63887, offset=0, option=None, proto=1, src='10.0.0.3', tos=0, total_length=84, ttl=64, version=4), icmp(code=0, csum=64290, data=echo(data=b'\x7f\x00\x00\x00:\x05\x01\x00\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#%&\'()*+,-./01234567', id=26282, seq=197), type=8)
[FW][INFO] dpid=0000000000000001: Blocked packet = ethernet(dst='4a:4f:e1:10:4d:b3', ethertype=2048, src='4a:64:c7:14:5e:f0'), ipv4(csum=12257, dst='10.0.0.1', flags=2, header_length=5, identification=63172, offset=0, option=None, proto=1, src='10.0.0.3', tos=0, total_length=84, ttl=64, version=4), icmp(code=0, csum=22468, data=echo(data=b'\x00s?f\x00\x00\x00:\x05\x01\x00\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#%&\'()*+,-./01234567', id=26282, seq=198), type=8)

```

`script /block.sh` pada directory `ips` agar serangan dapat diblokir. Gambar 3.31 menunjukkan `script block.sh` yang dijalankan berhasil memblokir IP penyerang dimana terdapat informasi IP penyerang, IP yang diserang, tipe ancaman atau serangan waktu masuk serangan dan waktu blokir. Saat `script block.sh` dijalankan, `block.sh` akan mengambil data `alert` terakhir penyerang dari log `alerts.csv` untuk teruskan ke `block.sh`. Scrip `block.sh` berisikan perintah `firewall` dari `controller ryu`. Maka `controller ryu` akan menginput IP penyerang pada `rules` dengan action `DENY` yang artinya apabila IP tersebut melakukan akses pada `server`, maka akan langsung didrop atau di blok oleh `Ryu`. Pada Gambar 3.32 dapat dilihat `Ryu` melakukan pemblokiran IP host 3. Dimana akses pada host 3 ke `server` ataupun sebaliknya akan langsung di blok oleh `Ryu`

Gambar 3.32 Ryu memblok IP penyerang

```
06/07-03:30:12.879948 , "Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:30:12.980307 , "Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:30:13.080659 , "Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:30:13.180950 , "Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:30:13.281331 , "Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:30:13.381558 , "Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:30:13.481734 , "Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:30:13.581968 , "Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:30:13.682336 , "Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:30:13.782695 , "Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:30:13.883055 , "Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:30:13.983438 , "Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:30:14.084047 , "Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:30:14.184422 , "Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:30:14.284790 , "Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:30:14.385177 , "Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:30:14.485576 , "Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:30:14.585953 , "Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:30:14.686320 , "Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:30:14.786509 , "Possible DDOS Attack Type : SYNflood",10.0.0.3,10.0.0.1
06/07-03:31:01.945424 , "Aktivasi Normal",fe80::4097:5bff:fed2:f850,ff02::2
06/07-03:31:55.193485 , "Aktivasi Normal",fe80::6d:63ff:feb0:27d0,ff02::2
```

Gambar 3.33 Alert pada Log Alerts.csv

Selanjutnya pada gambar 3.33 Dapat dilihat *alert* yang mendeteksi TCP SYN *Flood* telah berhenti. Ini menandakan serangan TCP SYN *Flood* telah berhasil diblokir.

```

"Node: server"
-----
Server listening on 5201
-----
Accepted connection from 10.0.0.2, port 56854
[ 7] local 10.0.0.1 port 5201 connected to 10.0.0.2 port 56866
[ ID] Interval      Transfer      Bitrate
[ 7] 0.00-1.00    sec 323 MBytes  2.71 Gbits/sec
[ 7] 1.00-2.00    sec 410 MBytes  3.44 Gbits/sec
[ 7] 2.00-3.00    sec 408 MBytes  3.42 Gbits/sec
[ 7] 3.00-4.00    sec 704 MBytes  5.91 Gbits/sec
[ 7] 4.00-5.00    sec 749 MBytes  6.28 Gbits/sec
[ 7] 5.00-6.00    sec 726 MBytes  6.09 Gbits/sec
[ 7] 6.00-7.00    sec 495 MBytes  4.15 Gbits/sec
[ 7] 7.00-8.00    sec 308 MBytes  2.58 Gbits/sec
[ 7] 8.00-9.00    sec 320 MBytes  2.69 Gbits/sec
[ 7] 9.00-10.00   sec 318 MBytes  2.67 Gbits/sec
[ 7] 10.00-10.04  sec 12.2 MBytes 2.32 Gbits/sec
-----
[ ID] Interval      Transfer      Bitrate
[ 7] 0.00-10.04   sec 4.66 GBytes 3.99 Gbits/sec
-----
Server listening on 5201

"host: h2"
-----
[ 7] 0.00-10.05   sec 3.88 GBytes 3.31 Gbits/sec
-----
iperf Done.
root@wenny-Lenovo-ideapad-110-14AST:/home/wenny/ips# iperf3 -c 10.0.0.1
Connecting to host 10.0.0.1, port 5201
[ 7] local 10.0.0.2 port 56866 connected to 10.0.0.1 port 5201
[ ID] Interval      Transfer      Bitrate      Retr  Cwnd
[ 7] 0.00-1.00    sec 341 MBytes  2.85 Gbits/sec  0    680 KBytes
[ 7] 1.00-2.00    sec 424 MBytes  3.56 Gbits/sec  90    870 KBytes
[ 7] 2.00-3.00    sec 409 MBytes  3.43 Gbits/sec  0    870 KBytes
[ 7] 3.00-4.00    sec 700 MBytes  5.87 Gbits/sec  0    875 KBytes
[ 7] 4.00-5.00    sec 742 MBytes  6.23 Gbits/sec  246   840 KBytes
[ 7] 5.00-6.00    sec 732 MBytes  6.14 Gbits/sec  0    840 KBytes
[ 7] 6.00-7.00    sec 494 MBytes  4.15 Gbits/sec  0    841 KBytes
[ 7] 7.00-8.01    sec 285 MBytes  2.37 Gbits/sec  0    841 KBytes
[ 7] 8.01-9.01    sec 326 MBytes  2.73 Gbits/sec  0    841 KBytes
[ 7] 9.01-10.00   sec 320 MBytes  2.70 Gbits/sec  0    841 KBytes
-----
[ ID] Interval      Transfer      Bitrate      Retr
[ 7] 0.00-10.00   sec 4.66 GBytes 4.00 Gbits/sec 336
[ 7] 0.00-10.04   sec 4.66 GBytes 3.99 Gbits/sec
-----
iperf Done.

```

Gambar 3.34 Hasil bandwidth dan transfer setelah blokir serangan

Setelah rest *firewall* pada *ryu* berhasil memblokir serangan . Terlihat nilai bandwidth mulai naik menjadi 3,99 Gbps dan untuk transfer di 4,66 GBps seperti yang ditunjukkan pada gambar 3.34. Kemudian dilakukan percobaan pada host 3 dengan melakukan ping IP ke *server*, terlihat bahwa host 3 sudah tidak dapat melakukan ping

```

"host: attacker1"
-----
root@wenny-Lenovo-ideapad-110-14AST:/home/wenny/ips# hping3 -c 1000 -d 120 -S -
-w 64 -p 80 --flood 10.0.0.1
HPING 10.0.0.1 (attacker1-eth0 10.0.0.1): S set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
^C
--- 10.0.0.1 hping statistic ---
9117011 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@wenny-Lenovo-ideapad-110-14AST:/home/wenny/ips# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.

```

IP kembali. Ditunjukkan pada Gambar 3.35

Gambar 3.35 Host 3 gagal melakukan ping ke server

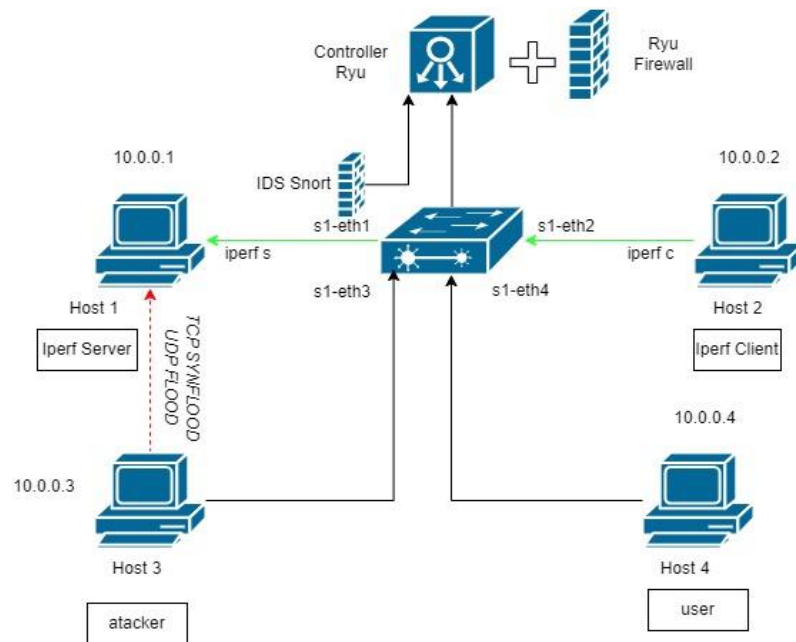
Dari pengujian sistem keamanan IPS yang telah dilakukan dapat dilihat bahwa sistem dapat berjalan dengan baik dalam mendeteksi dan memblokir serangan yang masuk, baik itu TCP SYN *Flood* maupun UDP *Flood*. Selanjutnya pada penelitian ini akan dilakukan pengujian lebih lanjut terhadap serangan dan sistem keamanan IPS dengan memvariasikan paket serangan dan melakukan pengambilan data nilai QoS berupa *throughput*, *CPU usage* dan *Memory usage* sehingga dapat melihat perbandingan kinerja jaringan dan sistem operasi sebelum di serang, saat dilakukan serangan dan saat diblokir serangan.

3.8. SKENARIO PENGUJIAN DAN PENGAMBILAN DATA

Tahap selanjutnya adalah pengambilan data dari skenario pengujian yang akan dilakukan. Skenario yang pertama adalah melakukan pengukuran QoS berupa *throughput* menggunakan *iperf*. Kemudian untuk skenario kedua adalah pengambilan data penggunaan *CPU Usage* dan *Memory Usage* menggunakan aplikasi *top*. Dalam hal ini dibuat perbandingan antara data yang diperoleh sebelum serangan, saat dilakukan serangan dan setelah dilakukan blokir serangan menggunakan sistem IPS.

3.8.1. Data *Quality Of Service* (QoS)

Pada tahap ini akan dilakukan pengambilan data QoS *throughput* menggunakan *Iperf*. Pengukuran Troughput melalui *iperf* ini bertujuan untuk menguji kecepatan data transfer jaringan SDN berupa nilai bandwidth dan nilai transfer sebelum dilakukan serangan, saat serangan tanpa integrasi IPS dan saat serangan blokir menggunakan sistem IPS. Berikut adalah topologi yang akan dijalankan pada skenario ini.



Gambar 3.36 Topologi pengambilan data QoS Throughput

Gambar 3.36 melakukan pengujian terhadap 3 kondisi. Untuk pengambilan data ini menggunakan *iperf server* dan *iperf client*. Host 1 bertindak sebagai *server* yang akan menjalankan *iperf server* dan host 2 menjalankan *iperf mode client* yang menguji kecepatan data transfer pada 3 kondisi tersebut. Pengambilan data pertama adalah saat *server* dalam keadaan normal. Dimana host 2 menjalankan *iperf c* menuju IP *server*. Setelah didapatkan hasil saat kondisi normal. Selanjutnya Host 3 akan melakukan serangan DDoS tanpa IPS dengan mengirimkan paket TCP SYN Flood dan UDP Flood. Dimana serangan akan dijalankan dengan variasi jumlah paket yang ditingkatkan secara bertahap yaitu 10, 100, 1000 dan 10000. Paket tersebut akan dikirimkan per detik dengan interval waktu pengambilan data selama 10 detik. Saat serangan sedang berlangsung tanpa IPS, host 2 akan melakukan pengujian kecepatan transfer kembali ke *server* dengan menjalankan *iperf* menuju *server*. Setiap variasi paket yang ditingkatkan maka akan diambil nilai throughput berupa transfer dan bandwidth melalui *iperf* tersebut.

Setelah didapatkan hasil saat serangan tanpa IPS, selanjutnya serangan yang berlangsung diintegrasikan oleh IPS sehingga IP penyerang diblokir oleh *Ryu* dan

aktivitas penyerang ke *server* otomatis akan terhenti. Pada kondisi ini host 2 akan dijalankan *iperf* kembali menuju *server* untuk melihat hasil *throughput* pada *server* saat serangan telah diblokir oleh IPS. Setelah semua data telah didapatkan maka selanjutnya membandingkan hasil data yang diperoleh dari 3 kondisi tersebut.

3.8.2. Data CPU Usage Dan Memory Usage

Skenario kedua yaitu pengambilan data penggunaan CPU dan Memori. Pengujian ini akan dilakukan menggunakan *software top* dimana pengambilan data pada skenario ini hanya melancarkan serangan pada 10000 serangan saja. Yaitu saat serangan TCP SYN *Flood* dan UDP *Flood*. Pengujian ini bertujuan untuk mengetahui dampak penggunaan CPU dan Memori pada jaringan SDN dalam kondisi normal, saat diserang tanpa IPS maupun saat serangan diintegrasikan IPS.