

## BAB IV

### HASIL DAN PEMBAHASAN

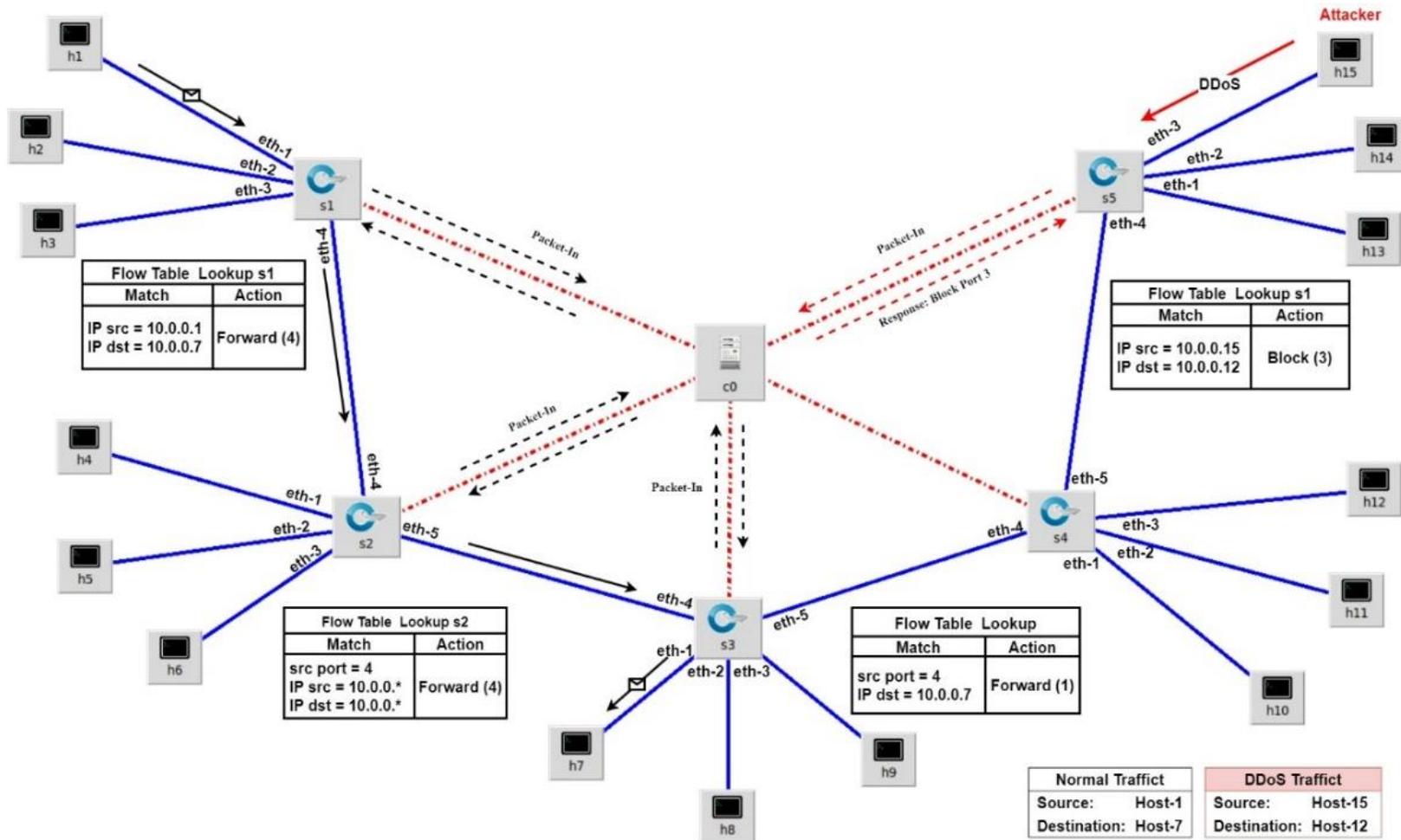
Pada bab ini, membahas mengenai analisis dan hasil dari penelitian yang dilakukan untuk mendeteksi dan memitigasi serangan *Distributed Denial of Service* (DDoS) pada jaringan *Software Defined Network* (SDN) menggunakan algoritma *Multilayer Perceptron* (MLP). Penelitian ini mencakup eksperimen yang dilakukan menggunakan Mininet sebagai emulator jaringan, Ryu *controller* sebagai pengendali jaringan, dan sFlow RT sebagai alat pemantau lalu lintas jaringan. Analisis dilakukan dengan melihat performa jaringan melalui pengukuran *delay*, *throughput*, *packet loss* dan sumber daya sistem.

#### 4.1. Analisis Jaringan

Topologi pada penelitian ini mencakup berbagai *node* dan *switch* sehingga dapat meniru skenario jaringan yang kompleks. Jaringan yang digunakan dalam penelitian ini adalah *traffic generator*, yang memungkinkan pengukuran nilai *delay*, *throughput*, dan *packet loss*. Berikut adalah gambar topologi dalam simulasi jaringan yang digunakan

*Traffic generator* digunakan untuk menghasilkan lalu lintas jaringan yang bervariasi sehingga dapat mensimulasikan berbagai jenis serangan DDoS. Berdasarkan hasil *traffic generator* tersebut dapat peneliti dapat mengukur respon jaringan terhadap serangan tersebut. Pengukuran performa jaringan ini penting untuk mengevaluasi seberapa efektif algoritma MLP dalam mendeteksi dan memitigasi serangan DDoS. Setiap metrik pengukuran dapat memberikan gambaran tentang bagaimana serangan mempengaruhi kinerja jaringan dan seberapa baik mitigasi yang diterapkan dapat memperbaiki performa jaringan.

Selain itu, topologi jaringan yang digunakan juga dirancang agar cukup fleksibel untuk diuji dengan berbagai skenario serangan, baik serangan yang bersifat volumetrik maupun serangan yang lebih canggih, seperti serangan berbasis protokol. Fleksibilitas ini memungkinkan peneliti untuk menilai efektivitas mitigasi dalam berbagai kondisi jaringan yang berbeda. Berikut adalah gambar topologi dalam simulasi jaringan yang digunakan.



Gambar 4.1 Topologi jaringan linear

Gambar 4.1 menunjukkan simulasi topologi jaringan SDN yang digunakan dalam penelitian ini. Terdapat tiga komponen utama dalam pembentukan jaringan SDN yaitu *host*, *OVS Switch*, dan *controller*. Setiap *switch* dikendalikan oleh *controller remote* yang berjalan pada alamat 127.0.0.1 di *port* 6653. *Switch* yang digunakan dalam penelitian ini berfungsi sebagai pengatur lalu lintas data dan pembuat aturan *flow* menggunakan *Ryu controller* dengan protokol *OpenFlow* 1.3.

Setiap *host* diatur dengan alamat IP dan MAC yang spesifik. Misalnya, *host* h1 memiliki IP 10.0.0.1/24 dan MAC address 00:00:00:00:00:01, sedangkan *host* h2 memiliki IP 10.0.0.2/24 dan MAC address 00:00:00:00:00:02. Pengaturan ini memungkinkan pengujian lalu lintas data yang terstruktur dan terkontrol antara *host-host* dalam jaringan. Berikut merupakan gambar *port* yang terhubung di masing-masing *node* pada topologi.

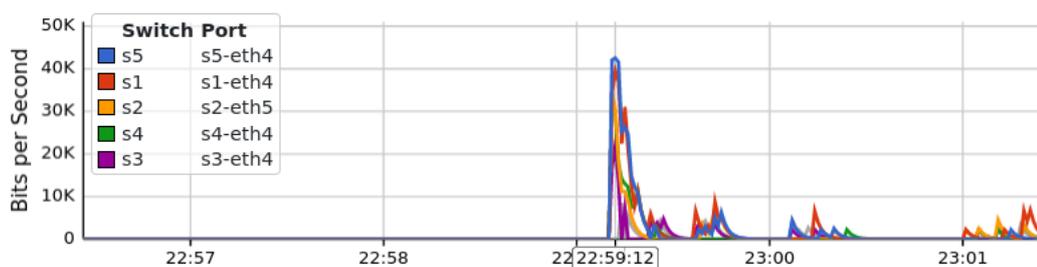
```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
h4 h4-eth0:s2-eth1
h5 h5-eth0:s2-eth2
h6 h6-eth0:s2-eth3
h7 h7-eth0:s3-eth1
h8 h8-eth0:s3-eth2
h9 h9-eth0:s3-eth3
h10 h10-eth0:s4-eth1
h11 h11-eth0:s4-eth2
h12 h12-eth0:s4-eth3
h13 h13-eth0:s5-eth1
h14 h14-eth0:s5-eth2
h15 h15-eth0:s5-eth3
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:h3-eth0 s1-eth4:s2-eth4
s2 lo: s2-eth1:h4-eth0 s2-eth2:h5-eth0 s2-eth3:h6-eth0 s2-eth4:s1-eth4 s2-eth5:
s3-eth4
s3 lo: s3-eth1:h7-eth0 s3-eth2:h8-eth0 s3-eth3:h9-eth0 s3-eth4:s2-eth5 s3-eth5:
s4-eth4
s4 lo: s4-eth1:h10-eth0 s4-eth2:h11-eth0 s4-eth3:h12-eth0 s4-eth4:s3-eth5 s4-eth5:
h5:s5-eth4
```

Gambar 4.2 Koneksi antara *host* dan *switch*

Gambar 4.2 adalah hasil dari perintah `net` pada terminal Mininet yang menunjukkan topologi jaringan dan koneksi antara *host* dan *switch*. Dalam topologi ini, ada lima *switch* yaitu s1 hingga s5 dan lima belas *host* yaitu h1 hingga h15. Setiap *host* terhubung ke sebuah *switch* melalui *port* tertentu. Misalnya, *host* h1 terhubung ke *switch* s1 pada *port* s1-eth1, *host* h2 ke s1 pada *port* s1-eth2, dan seterusnya. Selain itu, *switch* juga saling terhubung satu sama lain, membentuk jaringan yang lebih kompleks. *Switch* s1 terhubung ke *switch* s2 dan s4, *switch* s2 terhubung ke s1 dan s3, *switch* s3 terhubung ke s2 dan s4, *switch* s4 terhubung ke

s3 dan s5, dan *switch* s5 terhubung ke s4. Topologi ini memungkinkan adanya beberapa jalur antara *host*, yang meningkatkan redundansi dan ketahanan jaringan.

Saat jaringan pertama kali terhubung, *switch* dikendalikan oleh aplikasi `SimpleSwitch13` yang berjalan pada *Ryu controller*. *Switch* diinisialisasi untuk mengirimkan semua paket yang tidak dikenali ke *controller* melalui aturan default yang ditetapkan pada event `EventOFPSwitchFeatures`. Ketika paket diterima oleh *switch*, handler `EventOFPPacketIn` mempelajari alamat MAC sumber dan memetakan *port*-nya, serta menentukan *port* tujuan berdasarkan alamat MAC tujuan. Jika alamat tujuan diketahui, *flow* baru diinstal untuk menangani lalu lintas serupa di masa depan tanpa perlu mengirimkan paket ke *controller* lagi sehingga meningkatkan efisiensi jaringan. *Flow* diinstal dengan mempertimbangkan protokol yang berbeda seperti ICMP, TCP, dan UDP. Berikut merupakan gambar aktivitas *port* ketika *switch* pertama kali terhubung kedalam jaringan.



Gambar 4.3 Aktivitas *port* saat pertama kali terhubung

Gambar 4.3 menunjukkan aktivitas *port* pada *switch* di jaringan saat pertama kali terhubung. Grafik tersebut memvisualisasikan bit per detik untuk berbagai *port* pada *switch* yang berbeda. Pada awal koneksi, terlihat adanya lonjakan aktivitas lalu lintas jaringan yang signifikan. Ini menandakan bahwa ada volume data yang tinggi melewati *port-port* tersebut ketika jaringan mulai beroperasi. Lonjakan ini disebabkan oleh proses inisialisasi dan pengaturan *flow* pada *switch* oleh *Ryu controller*. Setelah lonjakan awal, aktivitas lalu lintas cenderung menurun dan stabil dengan beberapa fluktuasi kecil, yang menunjukkan bahwa jaringan mulai beroperasi dalam kondisi normal setelah proses inisialisasi selesai.

Selanjutnya fungsi *pingall* digunakan untuk melakukan pengecekan konektivitas antar seluruh *host* yang ada dalam jaringan. Cara kerja *pingall* yaitu dengan mendapatkan daftar semua *host* dan kemudian mengirimkan paket *Internet Control Message Protocol* (ICMP) dari satu *host* ke *host* lainnya untuk memastikan bahwa *host* tujuan dapat menerima dan merespons paket tersebut. Berikut merupakan hasil pada terminal mininet.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12 h13 h14 h15
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12 h13 h14 h15
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12 h13 h14 h15
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11 h12 h13 h14 h15
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11 h12 h13 h14 h15
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h12 h13 h14 h15
h12 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h13 h14 h15
h13 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h14 h15
h14 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h15
h15 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14
*** Results: 0% dropped (210/210 received)
```

Gambar 4.4 Hasil pengujian *pingall* pada terminal mininet

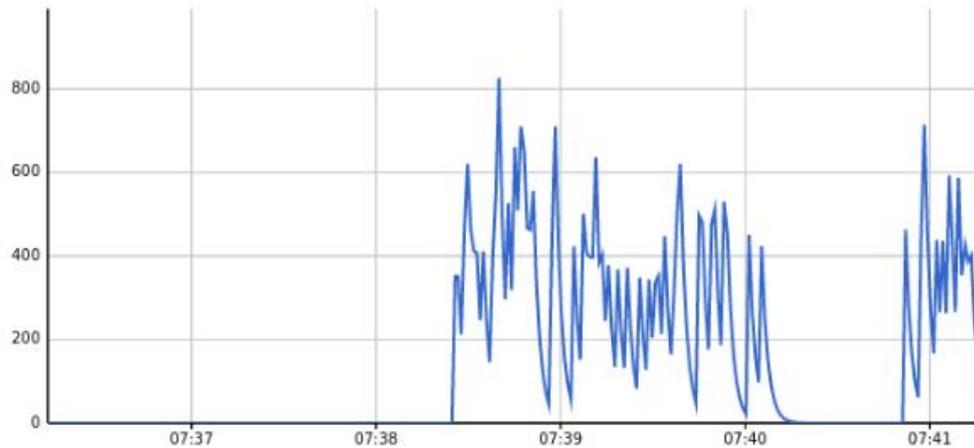
Gambar 4.4 menunjukkan hasil dari pengujian *pingall*, di mana setiap *host* mengirimkan paket ICMP ke semua *host* lainnya. Hasil ini memastikan bahwa semua *host* dan *switch* berfungsi dengan baik dalam jaringan. Nilai *drop packet* sebesar 0% menandakan tidak ada masalah konektivitas, seperti *host* yang tidak dapat dijangkau atau adanya *delay* tinggi dalam pengiriman paket. Dengan hasil ini, dapat dipastikan bahwa topologi jaringan telah terkonfigurasi dengan benar dan setiap perangkat dalam jaringan dapat berkomunikasi secara efektif tanpa ada hambatan.

## 4.2. Analisis Hasil Deteksi dan Mitigasi

### 4.2.1 Lalu lintas Normal

Pada kondisi lalu lintas normal, model *Multilayer Perceptron* (MLP) digunakan untuk memantau dan mendeteksi anomali dalam jaringan. Untuk

mengecek respon sistem terhadap lalu lintas sah digunakan perintah *ping*. Berikut merupakan hasil monitoring sistem ketika di lalui lalu lintas normal.



Gambar 4.5 Hasil monitoring lalu lintas normal

Gambar 4.5 merupakan grafik jumlah aliran data yang mengalir pada sistem dengan satuan Bits/s. Gambar di atas menunjukkan karakteristik lalu lintas normal dengan *traffic generator* yaitu *ping*, menghasilkan lalu lintas data sebanyak 800 Bits/s. Pada gambar di atas juga terlihat bahwa volume trafik stabil dan konsisten tanpa adanya lonjakan yang signifikan. Berikut merupakan hasil deteksi pada saat dilewati oleh lalu lintas normal atau lalu lintas sah.

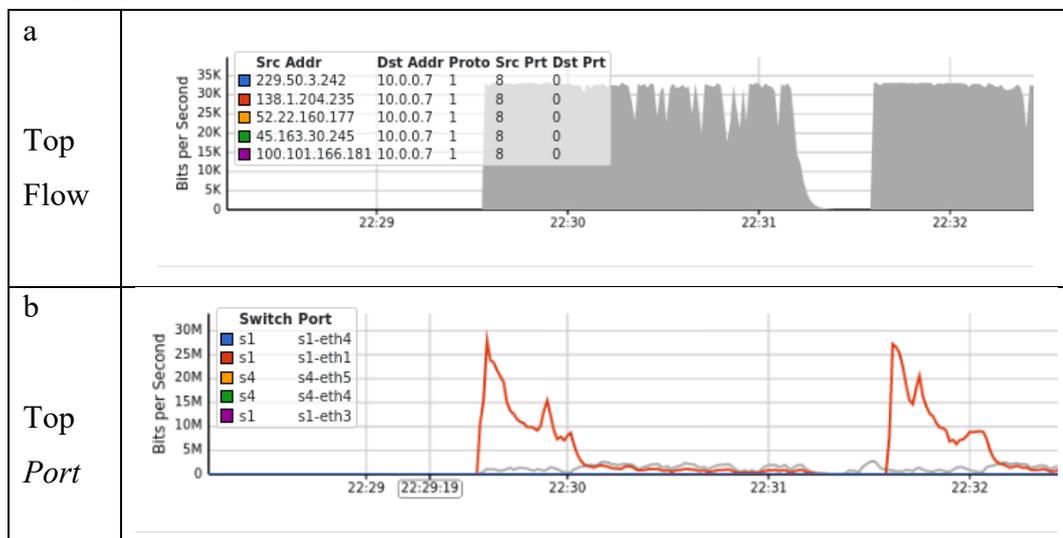
```
Traffic is mostly Normal.
-----
Traffic Analysis:
Normal traffic count: 264
DDoS traffic count: 0
Total traffic analyzed: 264
Traffic is mostly Normal.
-----
Traffic Analysis:
Normal traffic count: 2
DDoS traffic count: 0
Total traffic analyzed: 2
Traffic is mostly Normal.
-----
Traffic Analysis:
Normal traffic count: 2
DDoS traffic count: 0
Total traffic analyzed: 2
Traffic is mostly Normal.
-----
```

Gambar 4.6 Hasil deteksi lalu lintas normal

Berdasarkan Gambar 4.6, hasil deteksi menunjukkan bahwa MLP mampu dengan efektif mengenali pola lalu lintas yang normal dan stabil, tanpa memberikan *false positives*. Ini menunjukkan bahwa model MLP terlatih dengan baik untuk membedakan antara lalu lintas yang wajar dan pola yang mungkin mencurigakan.

#### 4.2.2 ICMP flood

Dalam eksperimen ini, serangan ICMP *flood* diinisiasi menggunakan perintah *hping3* untuk mengirimkan paket ICMP secara masif ke target. Perintah *hping3* dilakukan dari *host 1* menuju *host 7* dengan mengirimkan alamat IP sumber yang acak untuk setiap paket yang dikirim. Berikut merupakan hasil monitoring jaringan pada saat sistem di lewati oleh hasil simulasi ICMP *flood*.



Gambar 4.7 Hasil monitoring lalu lintas ICMP *flood*, (a) Hasil analisis flow saat ICMP *flood*, (b) Hasil analisis port saat ICMP *flood*

Gambar 4.7 menunjukkan karakteristik lalu lintas ICMP *flood* dalam hal jumlah bits setiap *flow* dan jumlah bits di setiap *port*. Pada grafik *Top Flows*, terlihat bahwa terdapat lonjakan volume trafik yang sangat tinggi mencapai 30 juta Bits/s, menandakan adanya serangan ICMP *flood* dengan bits per second yang meningkat tajam hingga mencapai puncaknya dengan tujuan menghabiskan *bandwidth* dan sumber daya jaringan. Adapun grafik *Top Ports* memperlihatkan aktivitas *port* selama serangan, dengan peningkatan signifikan pada jumlah bits per second di *port* S1 yang merupakan sumber ICMP *flood*, mencerminkan dampak serangan pada *throughput* jaringan. Serangan ICMP *flood* sering menggunakan sumber lalu lintas

yang acak, yaitu berbagai alamat IP sumber yang berbeda-beda, untuk menghindari deteksi dan pemblokiran mudah. Ini ditunjukkan dalam grafik *top flows* di mana terdapat beberapa alamat IP yang berkontribusi pada volume trafik yang tinggi. Respon sistem terhadap *traffic ICMP flood* ditunjukkan pada gambar 4.8.

```
Blocking port 1 in switch 1
-----
Traffic Analysis:
Legitimate Traffic = 0
DoS Traffic = 470
Total Traffic analyzed: 470
DDoS traffic detected.
Victim Host: h7
Mitigation process in progress!
-----
```

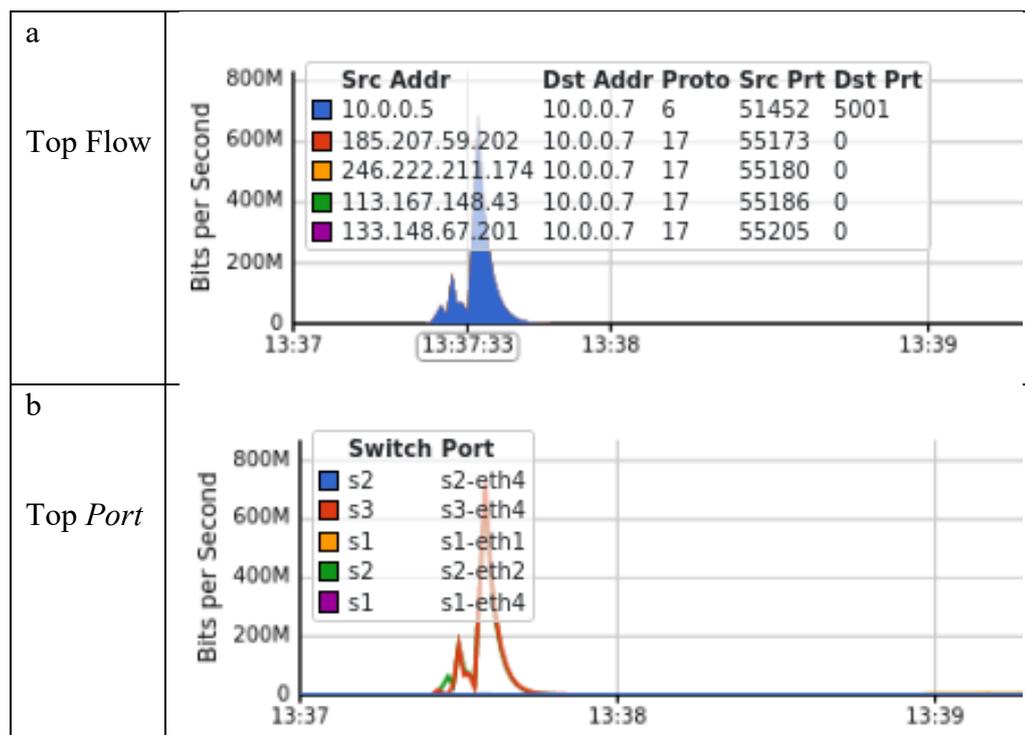
Gambar 4.8 Hasil deteksi dan mitigasi lalu lintas ICMP *flood*

Gambar 4.8 Merupakan hasil prediksi yang ditampilkan pada terminal, MLP berhasil mendeteksi serangan ICMP *flood* dengan akurasi yang tinggi. Hasil analisis menunjukkan bahwa tidak ada lalu lintas normal yang terdeteksi selama serangan. Sebagai gantinya, semua paket yang dianalisis diklasifikasikan sebagai lalu lintas DDoS. Selain itu, MLP berhasil mengidentifikasi *host* yang menjadi potensi korban serangan, yaitu h7, dalam setiap analisis. Ini menunjukkan bahwa model tidak hanya mampu mendeteksi serangan tetapi juga mengidentifikasi target yang diserang.

Berdasarkan hasil deteksi, sistem kemudian melakukan mitigasi terhadap paket ICMP *flood* dengan cara memblokir *port* sumber serangan. Pada gambar 4.8, menunjukkan *port* yang di blokir yaitu *port* 1 pada *switch* 1. Hal ini sesuai dengan perintah *traffic generator* yang menjalankan *hping3* dari *host* 1 menuju *host* 7.

### 4.2.3 UDP flood

Dalam eksperimen ini, serangan UDP *flood* dicirikan oleh pengiriman jumlah besar paket UDP ke *port* acak, yang dilakukan menggunakan *traffic generator* yaitu *hping3* yang dijalankan dari *host* 1 menuju *host* 7. Berikut merupakan karakteristik lalu lintas UDP *flood* saat melawati sistem deteksi DDoS pada jaringan SDN.



Gambar 4.9 Hasil monitoring lalu lintas UDP *flood*, (a) Hasil analisis flow saat UDP *flood*, (b) Hasil analisis *port* saat UDP *flood*

Gambar 4.9, menampilkan aktivitas *flow* UDP *flood* yang mengalir pada sistem. Pada grafik *Top Flows*, terlihat bahwa terdapat lonjakan volume trafik pada topologi jaringan yang mencapai 700 juta Bits/s, menandakan adanya serangan UDP *flood* dengan volume yang sangat tinggi. Tidak seperti TCP, UDP adalah protokol tanpa koneksi yang tidak memerlukan *handshake* tiga arah sehingga penyerang dapat mengirim paket dengan kecepatan tinggi tanpa menunggu respons dari target. Pada grafik *Top Port* juga mencatat aliran data yang mengalir pada setiap *port* di masing-masing *switch*. Aliran data tertinggi tercatat pada *switch* 3 dengan *port* ethernet 4. Trafik jaringan dengan jumlah yang besar ini mampu di deteksi sistem yang di tunjukan pada gambar 4.10

```

Blocking port 1 in switch 1
-----
Traffic Analysis:
Legitimate Traffic = 0
DoS Traffic = 526
Total Traffic analyzed: 526
DDoS traffic detected.
Victim Host: h7
Mitigation process in progress!
-----
Blocking port 1 in switch 1

```

Gambar 4.10 Hasil deteksi dan mitigasi lalu lintas UDP *flood*

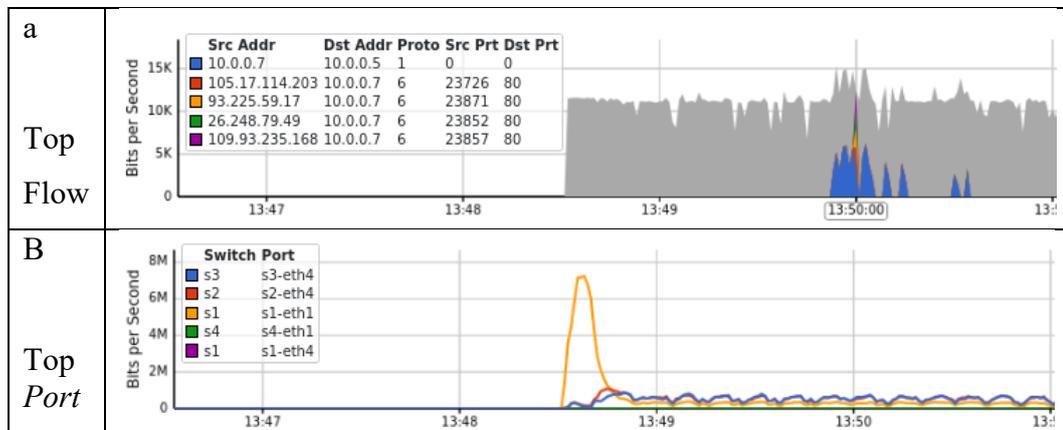
Gambar 4.10 Merupakan hasil deteksi dan mitigasi lalu lintas UDP *flood* yang ditampilkan pada terminal, MLP berhasil mendeteksi serangan UDP *flood* dengan akurasi tinggi. Hasil deteksi menunjukkan bahwa MLP dapat secara efektif membedakan antara lalu lintas normal dan lalu lintas serangan, dengan minimal kesalahan deteksi atau *false negative*. Dari 526 paket UDP *flood* yang di kirimkan, sistem menunjukkan bahwa tidak ada lalu lintas normal yang terdeteksi selama serangan. Semua paket yang dianalisis diklasifikasikan sebagai lalu lintas DDoS.

Pada sistem, berhasil mengidentifikasi *host 7* sebagai potensi korban serangan dalam setiap analisis. Selain itu terdapat respon sistem yang menunjukkan *port* dan *switch* yang di blokir yaitu pada *port 1* di *switch 1*. Hasil ini telah sesuai dengan perintah yang di kirimkan sebelumnya untuk membuat paket UDP *flood* yaitu perintah *hping3* dari *host 1* menuju *host 7*.

#### 4.2.4 TCP SYN *flood*

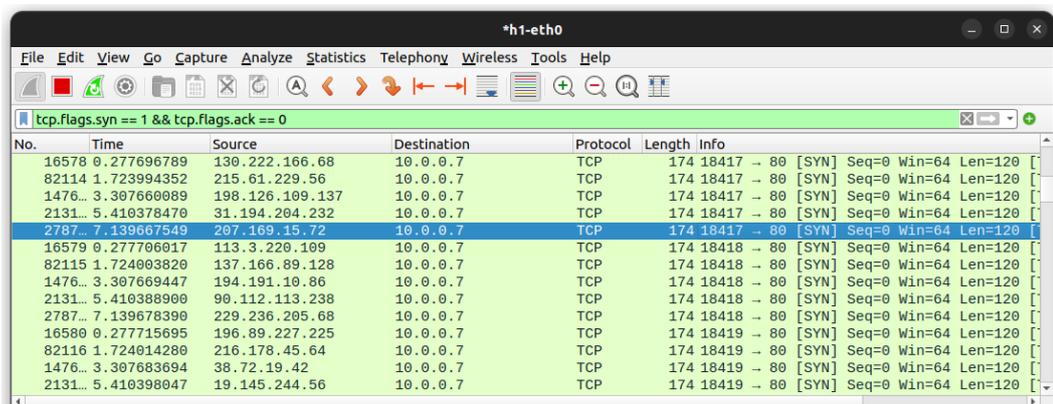
Serangan TCP SYN *flood* ditujukan untuk menghabiskan sumber daya server dengan membanjiri SYN *packets*. Deteksi dalam kasus ini dilakukan dengan mengamati peningkatan tajam dalam jumlah SYN *packets* yang tidak diikuti oleh ACK *packets*. Berikut merupakan gambar karakteristik TCP SYN *flood* yang di

ciptakan oleh *traffic generator Hping3* dari *host 1* menuju *host 7* dimana Penyerang mengirimkan sejumlah besar *SYN packets* ke server target.



Gambar 4.11 Hasil monitoring lalu lintas TCP SYN, (a) Hasil analisis flow saat TCP SYN flood, (b) Hasil analisis port saat TCP SYN flood

Gambar 4.11 merupakan hasil monitoring paket TCP SYN flood pada sistem. Pada grafik *Top Flows*, terlihat lonjakan volume trafik yang sangat tinggi mencapai 15 juta *Bits/s*. Adapun pada port *s1-eth1* yang merupakan sumber DDoS menampilkan jumlah aliran data sebanyak 7 juta *Bits/s* yang diakibatkan oleh sejumlah besar *SYN packets* yang dikirimkan oleh penyerang tanpa menyelesaikan proses *handshake* sehingga meninggalkan koneksi dalam status setengah terbuka dan memaksa server untuk mengalokasikan sumber daya untuk setiap koneksi yang tidak pernah diselesaikan. Untuk melihat *SYN packets* pada sistem digunakan Wireshark, yang di tunjukan pada gambar berikut.



Gambar 4.12 Lalu lintas TCP SYN flood pada Wireshark

Gambar 4.12 menunjukkan aplikasi Wireshark dengan filter “`tcp.flags.syn == 1 && tcp.flags.ack == 0`”, yang digunakan untuk menampilkan hanya paket TCP SYN tanpa bit ACK yang disetel. Hasil monitoring menunjukkan bahwa ada banyak koneksi yang tidak pernah diselesaikan yaitu paket SYN tanpa ACK. Hal ini dapat diidentifikasi dengan membandingkan jumlah paket SYN terhadap paket SYN-ACK dan ACK. Jumlah paket SYN yang jauh lebih tinggi dibandingkan dengan paket ACK menunjukkan adanya serangan TCP SYN *flood*.

```
Blocking port 1 in switch 1
-----
Traffic Analysis:
Legitimate Traffic = 0
DoS Traffic = 100
Total Traffic analyzed: 100
DDoS traffic detected.
Victim Host: h7
Mitigation process in progress!
```

Gambar 4.13 Hasil deteksi dan mitigasi lalu lintas TCP SYN *flood*

Berdasarkan Gambar 4.13, MLP menunjukkan kemampuan yang sangat baik dalam mengidentifikasi dan membedakan serangan ini dari lalu lintas TCP normal. Dari 100 paket yang di dikirimkan oleh *traffic generator*, semuanya terdeteksi sebagai DDoS. Dari hasil deteksi ini selanjutnya, sistem melakukan mitigasi kepada sumber *port* yang diklasifikasikan sebagai lalu lintas DDoS. Hasil ini menunjukkan bahwa sistem mampu untuk memblokir *port* 1 pada *switch* 1 yang merupakan sumber DDoS. Berdasarkan data ini, dapat disimpulkan bahwa MLP efektif dalam mengenali pola lalu lintas abnormal dan mendeteksi serangan TCP SYN *flood* dengan tepat, memastikan perlindungan yang lebih baik terhadap ancaman jaringan.

#### 4.2.5 Rata -Rata Efektivitas Deteksi

Tabel 4.1 merupakan hasil dari deteksi sistem. Tabel ini menjelaskan performa model *Multilayer Perceptron* (MLP) dalam mendeteksi serangan DDoS dan mengenali lalu lintas normal. *Confusion matrix* pada tabel ini menggambarkan hasil deteksi untuk empat jenis lalu lintas, yaitu lalu lintas normal, *ICMP flood*, *UDP flood*, dan *TCP SYN flood*.

Tabel 4.1 Hasil deteksi sistem

		Prediksi			
		Normal	ICMP <i>flood</i>	UDP <i>flood</i>	TCP SYN <i>flood</i>
Aktual	Normal	264	0	0	0
	ICMP <i>flood</i>	0	470	0	0
	UDP <i>flood</i>	0	0	526	0
	TCP SYN <i>flood</i>	0	0	0	100

Pada kondisi lalu lintas normal, model MLP berhasil mendeteksi seluruh paket dengan benar sebagai lalu lintas normal, menunjukkan True Positives (TP) sebesar 264 dan tidak ada False Positives (FP) atau *False Negatives* (FN). Ini menunjukkan bahwa model memiliki akurasi yang sempurna dalam mengenali lalu lintas yang wajar.

Dalam eksperimen *ICMP flood*, model MLP mendeteksi seluruh paket serangan sebagai DDoS dengan TP sebesar 470 dan tanpa FP atau FN. Hasil ini menunjukkan bahwa model sangat efektif dalam mengenali pola serangan *ICMP flood* dengan akurasi yang tinggi.

Pada serangan *UDP flood*, model MLP juga menunjukkan performa yang sangat baik dengan TP sebesar 526 dan tidak ada FP atau FN, yang menunjukkan bahwa model dapat secara efektif membedakan antara lalu lintas normal dan lalu lintas serangan *UDP flood*.

Serangan *TCP SYN flood* juga dideteksi dengan sempurna oleh model MLP, dengan TP sebesar 100 dan tanpa adanya FP atau FN. Ini menunjukkan kemampuan model dalam mengidentifikasi dan mengklasifikasikan serangan *TCP SYN flood* dengan akurasi yang tinggi.

Secara keseluruhan, hasil yang ditampilkan dalam Tabel 4.1 menunjukkan bahwa model MLP memiliki akurasi pengujian yang sangat baik mencapai 100% dalam mendeteksi serangan DDoS. Tidak adanya *false positives* dan *false negatives* menunjukkan keefektifan dan keandalan model dalam menjaga keamanan jaringan dengan mengklasifikasikan lalu lintas secara akurat. Selain itu model MLP tidak hanya efektif dalam mendeteksi berbagai jenis serangan DDoS tetapi juga dalam memastikan bahwa lalu lintas normal tidak salah dikenali sebagai ancaman.

Efektifitas hasil deteksi akan berpengaruh besar pada hasil mitigasi. Hal ini menunjukkan bahwa model dapat digunakan dalam lingkungan jaringan yang sibuk dan kompleks tanpa menyebabkan peringatan palsu yang dapat mengganggu operasi jaringan.

### 4.3. Analisis Performa Jaringan

Untuk mengukur performa jaringan dan efektifitas sistem dalam mendeteksi dan mengatasi serangan DDoS pada penelitian ini, digunakan beberapa parameter utama yaitu *delay*, *throughput*, *jitter*, dan *resource utilization*. Pengukuran dilakukan pada kondisi normal dan selama serangan DDoS untuk memahami dampak serangan terhadap performa jaringan.

#### 4.3.1 Jitter

*Jitter* adalah variasi waktu pengiriman paket dalam jaringan, yang merupakan salah satu parameter penting dalam mengukur kualitas layanan jaringan. *Jitter* diukur sebagai selisih antara waktu tiba dari paket-paket berturut-turut yang diterima. Berikut merupakan data hasil pengukuran *jitter* pada berbagai skenario jaringan.

```
-----
Client connecting to 10.0.0.2, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215,21 us (kalman adjust)
UDP buffer size: 208 KByte (default)
-----
[ 1] local 10.0.0.10 port 57098 connected with 10.0.0.2 port 5001
[ ID] Interval      Transfer      Bandwidth
[ 1] 0.0000-10.0158 sec 1.25 MBytes  1.05 Mbits/sec
[ 1] Sent 896 datagrams
[ 1] Server Report:
[ ID] Interval      Transfer      Bandwidth      Jitter    Lost/Total Datagrams
[ 1] 0.0000-10.0105 sec 1.25 MBytes  1.05 Mbits/sec  0.026 ms  0/895 (0%)
root@ivan-virtual-machine:~/Documents/DDoS Detection V3# █
```

Gambar 4.14 *Jitter* pada jaringan lalu lintas normal

Gambar 4.14 merupakan hasil pengukuran *jitter* pada saat dilalui lalu lintas normal. Pengukuran *jitter* pada kondisi normal menunjukkan nilai yang sangat rendah, yaitu 0.026 ms. Menurut standar TIPHON, nilai ini masuk dalam kategori "sangat bagus" karena berada di bawah 75 ms. Hal ini mencerminkan performa jaringan yang optimal tanpa adanya gangguan signifikan.

Pada skenario kedua, sistem diuji selama terjadinya serangan DDoS, sementara pada skenario ketiga, sistem diuji dengan paket DDoS namun disertai dengan proses mitigasi. Paket DDoS dihasilkan menggunakan *traffic generator hping3* dan dikirimkan dari H1 ke H7. Pengujian *jitter* dilakukan pada H2 untuk mengukur dampak dari serangan tersebut. Proses mitigasi dilakukan dengan memblokir *port* sumber DDoS, yaitu *port eth1* pada *switch* 1. Tujuan dari mitigasi ini adalah untuk menilai efektivitasnya dalam mengurangi dampak serangan DDoS pada jaringan. Berikut adalah hasil pengukuran *jitter* saat serangan DDoS terjadi dan setelah dilakukan mitigasi.

Tabel 4.2 *Jitter* selama serangan dan setelah mitigasi DDoS

<b>Jenis Lalu lintas DDoS</b>	<b><i>Jitter</i> Selama DDoS Terjadi (ms)</b>	<b><i>Jitter</i> Setelah Mitigasi (ms)</b>
ICMP <i>flood</i>	RTO	0.011
UDP <i>flood</i>	RTO	0.013
TCP SYN <i>flood</i>	RTO	0.007

Tabel 4.2 merupakan pengukuran *jitter* saat serangan DDoS terjadi, pengukuran tidak dapat dilakukan karena paket tidak dapat diterima atau biasa disebut *Request time out* (RTO). Hal ini disebabkan oleh kondisi jaringan yang tidak stabil akibat banyaknya paket yang membanjiri jaringan selama serangan DDoS. Kemacetan dan kelebihan beban jaringan yang dihasilkan oleh serangan tersebut mengakibatkan ketidakmampuan jaringan untuk menjaga interval waktu pengiriman paket yang konsisten.

Namun, setelah mitigasi diterapkan, *jitter* menunjukkan penurunan yang signifikan, mendekati kondisi normal. Pada serangan ICMP *flood*, *jitter* turun menjadi 0.011 ms, pada serangan UDP *flood* menjadi 0.013 ms, dan pada serangan TCP SYN *flood* menjadi 0.007 ms. Berdasarkan standar TIPHON, nilai ini masuk dalam kategori "sangat bagus" karena berada di bawah 75 ms. Mekanisme mitigasi

yang efektif berhasil mengurangi dampak serangan DDoS, mengembalikan stabilitas jaringan, dan memastikan bahwa paket data tiba dengan interval waktu yang lebih konsisten. Hasil pengukuran *jitter* setelah mitigasi menunjukkan nilai jauh lebih rendah dibandingkan dengan saat serangan berlangsung, menandakan bahwa langkah mitigasi yang diterapkan berhasil memulihkan performa jaringan, memastikan kualitas layanan yang optimal bagi aplikasi *real-time*.

#### 4.3.2 Delay

*Delay* menunjukkan waktu round-trip dari pengiriman permintaan hingga penerimaan respons. *Delay* jaringan diukur dengan menggunakan perintah "*ping*". Perintah "*ping*" ini mengirimkan paket ICMP Echo Request dan mengukur waktu yang dibutuhkan untuk menerima ICMP Echo Reply dari *host* tujuan. Berikut adalah contoh hasil dari perintah "*ping*" yang menunjukkan *delay* jaringan.

Tabel 4.3 *Delay* jaringan pada lalu lintas normal

<b>Delay Lalu lintas Normal</b>			
min	avg	max	<i>mdev</i>
0.052 ms	0.280 ms	8.492 ms	1.1 ms

Berdasarkan Tabel 4.3, pada lalu lintas data normal, *delay* rendah dan stabil, menunjukkan bahwa paket ICMP *echo request* dan *echo reply* mengalir dengan lancar. Nilai *delay* rata-rata pada lalu lintas jaringan normal menunjukkan angka 0.28 ms. Berdasarkan standar TIPHON, nilai ini masuk dalam kategori "sangat bagus" yang menandakan bahwa jaringan sehat dan tidak ada gangguan signifikan dalam jalur komunikasi antara *host*.

Selanjutnya, sistem diuji selama berlangsungnya serangan DDoS, dan sistem diuji dengan adanya paket DDoS namun disertai dengan proses mitigasi. Paket DDoS dihasilkan menggunakan *traffic generator hping3* dan dikirimkan dari H1 ke H7. Kemudian dilakukan pengukuran *delay* pada H2 untuk mengevaluasi dampak serangan tersebut. Pada proses mitigasi sistem akan memblokir *port* sumber DDoS dari hasil prediksi model MLP, yaitu *port eth1* pada *switch 1*. Berikut adalah hasil pengukuran *delay* saat serangan DDoS terjadi dan setelah mitigasi diterapkan.

Tabel 4.4 *Delay* selama serangan dan setelah mitigasi DDoS

Jenis Lalu lintas DDoS	Selama DDoS Terjadi (ms)				Setelah Mitigasi (ms)			
	min	avg	max	<i>mdev</i>	min	avg	max	<i>mdev</i>
ICMP <i>flood</i>	RTO	RTO	RTO	RTO	0.028	0.242	8.717	1.162
UDP <i>flood</i>	RTO	RTO	RTO	RTO	0.037	0.4	20.409	2.322
TCP SYN <i>flood</i>	RTO	RTO	RTO	RTO	0.04	0.375	17.544	2.045

Tabel 4.4 adalah hasil pengukuran *delay* pada dua skenario jaringan yaitu Selama DDoS Terjadi dan setelah mitigasi. Pada kondisi jaringan selama serangan DDoS. Pengukuran *delay* tidak dapat dilakukan karena kemacetan yang sangat parah dalam jaringan, yang mengakibatkan banyak paket ICMP Echo Request tidak mendapatkan respons ICMP Echo Reply dari *host* tujuan sehingga terjadi RTO. Ketidakmampuan untuk mengukur *delay* ini menunjukkan adanya kemacetan yang ekstrem dan ketidakstabilan dalam jaringan.

Setelah mitigasi, kondisi *delay* jaringan menunjukkan perubahan yang signifikan dibandingkan dengan saat serangan DDoS berlangsung. Untuk serangan ICMP *flood*, setelah mitigasi diterapkan, *delay* menunjukkan nilai minimum sebesar 0.028 ms, rata-rata 0.242 ms, maksimum 8.717 ms, dan deviasi standar (*mdev*) sebesar 1.162 ms. Angka-angka ini menunjukkan bahwa mekanisme mitigasi mampu mengembalikan *delay* jaringan ke tingkat yang mendekati kondisi normal, meskipun masih ada sedikit fluktuasi. Berdasarkan standar TIPHON, nilai rata-rata *delay* ini masuk dalam kategori "Sangat Bagus" karena berada di bawah 150 ms.

Pada serangan UDP *flood*, *delay* setelah mitigasi mencatat nilai minimum sebesar 0.037 ms, rata-rata 0.4 ms, maksimum 20.409 ms, dan deviasi standar (*mdev*) sebesar 2.322 ms. Ini menandakan bahwa mitigasi berhasil mengurangi dampak serangan, meskipun ada beberapa lonjakan dalam waktu *round-trip* yang mungkin disebabkan oleh sisa-sisa kemacetan lalu lintas DDoS. Menurut standar TIPHON, nilai rata-rata *delay* setelah mitigasi pada serangan UDP *flood* juga masih dalam kategori "Sangat Bagus".

Pada serangan TCP SYN *flood*, meskipun ada peningkatan dalam nilai maksimum, rata-rata *delay* dan deviasi standar menunjukkan adanya perbaikan signifikan dalam stabilitas jaringan setelah mitigasi. Nilai rata-rata *delay* setelah di

lakukan mitigasi pada serangan TCP SYN *flood* juga termasuk dalam kategori "Sangat Bagus" berdasarkan standar TIPHON.

#### 4.3.3 Throughput

*Throughput* jaringan dapat diukur dengan memeriksa jumlah data yang berhasil dikirimkan dalam satuan waktu tertentu. Untuk mengukur *throughput*, digunakan *iperf*, untuk mengukur kecepatan transmisi data antara dua *host*. Berikut adalah contoh hasil dari *iperf* yang menunjukkan *throughput*.

Tabel 4.5 *Throughput* jaringan pada lalu lintas normal

<b><i>Throughput</i> Lalu lintas Normal</b>		
Interval	Transfer	<i>Bandwidth</i>
0-10 s	6.24 Gbps	5.36 Gbps

Tabel 4.5 menunjukkan bagaimana mitigasi mempengaruhi jumlah data yang di transfer dan *bandwidth* pada berbagai jenis lalu lintas data. Pada lalu lintas normal menunjukkan bahwa dalam interval waktu 10 detik, sebanyak 6.24 Gbps data berhasil ditransfer dengan *bandwidth* sebesar 5.36 Gbits/sec. Menurut standar TIPHON, nilai ini menunjukkan performa jaringan yang masuk dalam kategori "sangat bagus" karena *throughput* yang tinggi. Hal ini menandakan bahwa jaringan berada dalam kondisi optimal sehingga mampu menangani volume data yang besar.

Selanjutnya, dilakukan pengukuran *throughput* selama serangan DDoS, dan setelah mitigasi diterapkan. Paket DDoS dari H1 ke H7 dihasilkan menggunakan *hping3* dan pengukuran *throughput* dilakukan pada H2. Mitigasi dilakukan berdasarkan prediksi model MLP dengan memblokir *port* sumber serangan DDoS yaitu *port eth1* pada *switch* 1.

Tabel 4.6 *Throughput* selama serangan dan setelah mitigasi DDoS

Jenis Lalu lintas DDoS	Interval	Selama DDoS Terjadi (Gbyts)		Setelah Mitigasi (Gbyts)	
		Transfer	<i>Bandwidth</i>	Transfer	<i>Bandwidth</i>
ICMP <i>flood</i>	0-10 s	RTO	RTO	3.75	3.21
UDP <i>flood</i>		RTO	RTO	3.61	3.09
TCP SYN <i>flood</i>		RTO	RTO	4.06	3.49

Tabel 4.6 Menunjukkan nilai *throughput* jaringan selama serangan DDoS berlangsung, *throughput* tidak dapat di ukur karena topologi jaringan tidak lagi dapat saling berkomunikasi, yang ditandai dengan pesan kesalahan *tcp connect failed* saat melakukan pengecekan nilai *throughput*. Penyebab rusaknya koneksi antar jaringan adalah beban jaringan yang berlebihan, yang menyebabkan penurunan efisiensi dalam transmisi data. Hal ini menunjukkan bahwa serangan DDoS dapat secara signifikan mengurangi kapasitas jaringan untuk mentransmisikan data.

Setelah mitigasi diterapkan, kondisi jaringan menunjukkan perbaikan yang signifikan pada berbagai jenis serangan DDoS. Pada serangan ICMP *flood*, transfer tercatat sebesar 3.75 *Gbytes* dengan *bandwidth* 3.21 *Gbits/sec*. Pada serangan UDP *flood*, setelah mitigasi, transfer mencapai 3.61 *Gbytes* dengan *bandwidth* sebesar 3.09 *Gbits/sec*. Untuk serangan TCP SYN *flood*, transfer setelah mitigasi tercatat sebesar 4.06 *Gbytes* dengan *bandwidth* 3.49 *Gbits/sec*. Nilai-nilai ini menunjukkan bahwa mitigasi berhasil mengurangi dampak serangan tetapi dan memungkinkan jaringan untuk berfungsi dengan kinerja yang hampir optimal. Menurut standar TIPHON, nilai ini masuk dalam kategori "sangat bagus" karena *throughput* jaringan mencapai *gigabit* per detik.

#### 4.3.4 Packet Loss

*Packet loss* adalah parameter penting yang digunakan untuk mengukur performa jaringan, terutama dalam kondisi normal dan saat terjadi serangan DDoS. *Packet loss* mengindikasikan persentase paket data yang hilang atau tidak berhasil mencapai tujuan selama proses transmisi. Parameter ini sangat penting untuk mengevaluasi keandalan dan stabilitas jaringan. Berikut adalah hasil pengukuran *packet loss* menggunakan perintah "*ping*".

Tabel 4.7 *Packet loss* pada lalu lintas normal

<b>Packet Loss Lalu lintas Normal</b>	
Jumlah Paket	<i>packet loss</i> (%)
100	0

Tabel 4.7 menunjukkan hasil pengukuran *packet loss* pada kondisi normal. Hasil pengukuran *packet loss* pada lalu lintas jaringan normal, menunjukkan tidak ada *packet loss* yang terjadi selama 100 pengiriman paket, yang menunjukkan bahwa jaringan beroperasi dalam kondisi optimal. Semua paket *ICMP echo request* yang dikirim oleh h1 diterima dan direspon oleh h7 tanpa adanya kehilangan paket. Ini mencerminkan performa jaringan yang stabil dan efisien. Menurut standar TIPHON, nilai ini masuk dalam kategori "sangat bagus" karena *packet loss* berada di bawah 1%. Ini mencerminkan performa jaringan yang stabil dan efisien.

Selanjutnya, dilakukan pengukuran *packet loss* selama terjadinya serangan DDoS dan setelah penerapan mitigasi. Paket DDoS dihasilkan menggunakan *hping3* dan dikirim dari H1 ke H7, sementara pengukuran *packet loss* dilakukan pada H2. Proses mitigasi dilakukan berdasarkan prediksi model MLP dengan memblokir *port* sumber serangan DDoS, yaitu *port eth1* pada *switch 1*.

Tabel 4.8 *Packet loss* selama serangan dan setelah mitigasi DDoS

Jenis Lalu lintas Data	Jumlah Paket	Selama DDoS Terjadi	Setelah Mitigasi
		<i>packet loss</i> (%)	<i>packet loss</i> (%)
ICMP <i>flood</i>	100	100	39
UDP <i>flood</i>		100	35
TCP SYN <i>flood</i>		100	49

Tabel 4.8 adalah data *packet loss* Selama DDoS Terjadi dan setelah mitigasi dilakukan, yang di tandai oleh tingkat *packet loss* meningkat secara signifikan selama serangan DDoS. Pada tabel di atas dapat diamati bahwa nilai *packet loss* pada saat serangan DDoS tanpa adanya proses mitigasi serangan menyebabkan kemacetan dalam jaringan, mengakibatkan banyak paket yang hilang dan tidak mencapai tujuan mereka. Tingkat *packet loss* yang tinggi menunjukkan bahwa serangan DDoS secara drastis mengganggu aliran data dalam jaringan, menurunkan efisiensi dan kinerja jaringan secara keseluruhan.

Selama serangan ICMP *flood*, *packet loss* Selama DDoS Terjadi adalah 100%, menunjukkan bahwa tidak ada paket yang berhasil mencapai tujuan karena kemacetan jaringan yang parah. Setelah mitigasi, *packet loss* turun menjadi 39%, menunjukkan perbaikan signifikan dalam performa jaringan meskipun belum sepenuhnya pulih. Menurut standar TIPHON, nilai ini masih dalam kategori "buruk" karena lebih dari 25%.

Pada serangan UDP *flood*, *packet loss* juga mencapai 100%. Setelah mitigasi, *packet loss* berkurang menjadi 35%, yang masih dalam kategori "buruk" menurut standar TIPHON, tetapi menunjukkan bahwa tindakan mitigasi efektif dalam mengurangi dampak serangan dan memungkinkan lebih banyak paket untuk berhasil dikirimkan.

Pada serangan TCP SYN *flood*, *packet loss* adalah 100%. Setelah mitigasi, *packet loss* berkurang menjadi 49%, menunjukkan bahwa meskipun masih ada kehilangan paket yang signifikan, jaringan menunjukkan perbaikan dalam kemampuannya untuk menangani lalu lintas data, meskipun nilai ini tetap dalam kategori "buruk" menurut standar TIPHON.

Secara keseluruhan, hasil pengukuran *packet loss* setelah mitigasi menunjukkan bahwa langkah-langkah mitigasi yang diterapkan efektif dalam mengurangi dampak serangan DDoS. Meskipun *packet loss* belum sepenuhnya kembali ke kondisi normal, terdapat perbaikan yang signifikan. Hal ini membuktikan bahwa strategi mitigasi yang diterapkan berhasil melindungi jaringan dari dampak negatif serangan DDoS. Menurut standar TIPHON, meskipun masih dalam kategori "buruk" mitigasi telah membawa peningkatan yang berarti dalam stabilitas dan efisiensi jaringan.

#### 4.3.5 *Resource Utilization*

*Resource utilization* adalah parameter penting untuk memahami seberapa efisien sistem menggunakan sumber dayanya, seperti CPU dan memori, dalam berbagai kondisi lalu lintas jaringan. Analisis *resource utilization* dilakukan untuk mengevaluasi performa sistem dalam kondisi normal dan selama serangan DDoS, serta untuk memastikan bahwa sistem dapat menangani beban yang tinggi tanpa mengalami penurunan performa yang signifikan.

Namun karena simulasi dilakukan secara virtual menggunakan Mininet sehingga memiliki keterbatasan dalam pembagian dan pemantauan sumber daya CPU ke setiap *host* virtual. *Host* dalam Mininet adalah *namespace* jaringan Linux, yang berarti mereka adalah entitas virtual yang berjalan di atas kernel Linux yang sama dengan sistem *host*. Meskipun mereka dapat memiliki alamat IP dan aturan jaringan mereka sendiri, mereka berbagi kernel yang sama dan sumber daya CPU

dengan sistem *host*. Sehingga penggunaan CPU yang terbaca mencerminkan penggunaan seluruh sistem, bukan *host* virtual individu. Namun pada penelitian ini tetap dilakukan pemantauan *resource* untuk melihat respon sistem terhadap berbagai kondisi.

Tabel 4.9 *Resource utilization* pada lalu lintas normal

Resource Lalu lintas Normal (%)		
CPU Sistem	Memory Usage	CPU Process
12.5	39.1	0.3

Tabel 4.9 merupakan hasil pemantauan *resource utilization* dalam kondisi normal menunjukkan bahwa sistem beroperasi dengan efisien, dengan penggunaan CPU dan memori yang stabil. CPU Process menunjukkan angka 0.3%, yang menandakan persentase penggunaan CPU dalam kondisi normal, nilai ini cenderung rendah. CPU Sistem menunjukkan angka 12.5%, yang merupakan persentase total penggunaan CPU sistem dalam kondisi normal, nilai ini juga relatif rendah. Memory Usage berada pada angka 39.1%, nilai ini stabil dan tidak menunjukkan lonjakan yang signifikan. Ini mencerminkan bahwa jaringan mampu menangani lalu lintas normal tanpa mengalami kelebihan beban.

Selanjutnya, dilakukan pengukuran *resource utilization* selama serangan DDoS dan setelah mitigasi diterapkan. Paket DDoS dikirim dari H1 ke H7 menggunakan *traffic generator hping3*, dan pemantauan pemanfaatan sumber daya dilakukan pada H2. Pengukuran mencakup penggunaan CPU, memori, dan *bandwidth* untuk menilai dampak serangan terhadap kinerja sistem. Mitigasi dilakukan berdasarkan prediksi model MLP dengan memblokir *port* sumber serangan DDoS, yaitu *port eth1* pada *switch* 1. Tujuan dari pengukuran ini adalah untuk menilai efektivitas mitigasi dalam mengurangi beban pada sumber daya jaringan dan memastikan bahwa sistem tetap berfungsi dengan baik meskipun terjadi serangan.

Tabel 4.10 *Resource utilization* pada lalu lintas DDoS

Jenis Lalu lintas DDoS	Resource Selama DDoS Terjadi (%)			Resource Setelah Mitigasi (%)		
	CPU Sistem	Memory Usage	CPU Process	CPU Sistem	Memory Usage	CPU Process

ICMP <i>flood</i>	99.2	52	1.9	74.7	8.56	11.1
UDP <i>flood</i>	100	59.8	3.37	53.6	51.2	0.39
TCP SYN <i>flood</i>	100	55.9	0.58	48.8	39.6	0.37

Tabel 4.10 menunjukkan penggunaan sumber daya CPU dan memori pada berbagai jenis lalu lintas data sebelum dan setelah mitigasi diterapkan. Berikut adalah penjelasan detail berdasarkan data yang ditunjukkan dalam tabel.

Selama serangan ICMP *flood*, mitigasi berhasil mengurangi penggunaan CPU sistem dari 99.2% menjadi 74.7%, dan penggunaan memori dari 52% menjadi 8.56%. Namun, penggunaan CPU oleh proses tertentu meningkat dari 1.9% menjadi 11.1%, menunjukkan bahwa meskipun beban keseluruhan berkurang, proses spesifik yang terkait dengan mitigasi harus bekerja lebih keras untuk menangani serangan.

Pada serangan UDP *flood*, penggunaan CPU sistem berkurang secara signifikan dari 100% menjadi 53.6%, dan penggunaan memori menurun dari 59.8% menjadi 51.2%. Penggunaan CPU oleh proses tertentu juga mengalami penurunan drastis dari 3.37% menjadi 0.39%, menunjukkan bahwa mitigasi tidak hanya mengurangi beban keseluruhan tetapi juga mengoptimalkan penggunaan sumber daya pada *level proses*.

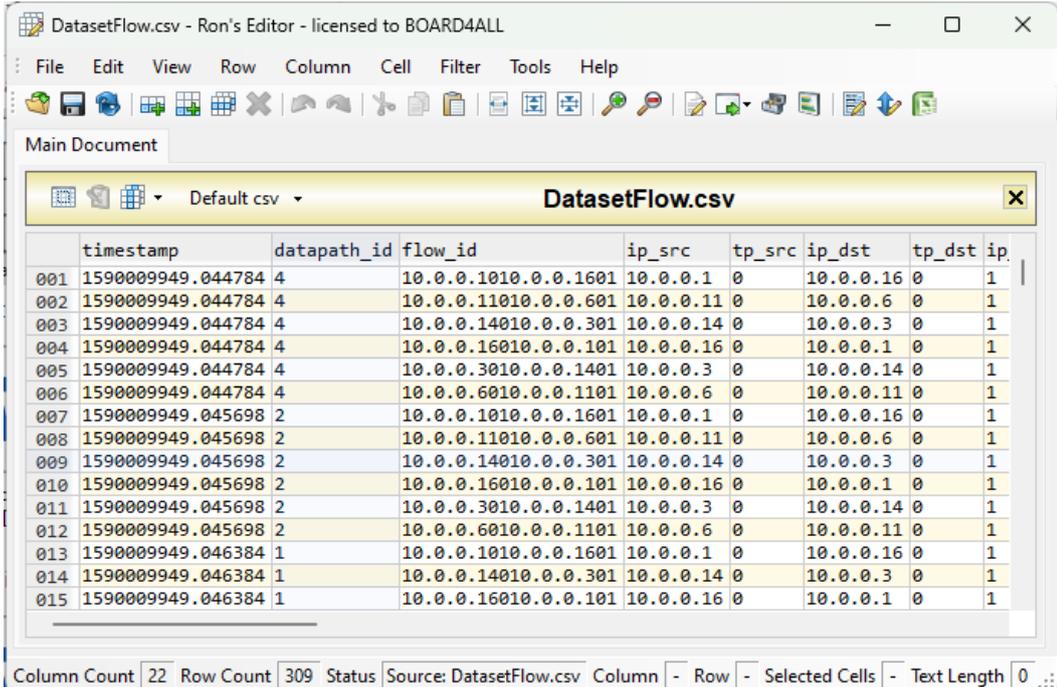
Untuk serangan TCP SYN *flood*, penggunaan CPU sistem turun dari 100% menjadi 48.8%, dan penggunaan memori berkurang dari 55.9% menjadi 39.6%. Penggunaan CPU oleh proses tertentu juga menurun dari 0.58% menjadi 0.37%, menunjukkan bahwa mitigasi berhasil mengurangi beban pada seluruh sistem dan memastikan bahwa sumber daya digunakan dengan lebih efisien.

Penurunan penggunaan sumber daya ini menunjukkan efektivitas langkah-langkah mitigasi dalam mengurangi dampak serangan DDoS, memulihkan performa jaringan, dan memastikan stabilitas serta efisiensi operasi jaringan. Penurunan penggunaan sumber daya ini dikarenakan sistem tidak lagi perlu menyimpan dan memproses informasi yang berlebihan terkait dengan lalu lintas berbahaya. Secara keseluruhan, hasil ini menunjukkan bahwa tindakan mitigasi yang diterapkan efektif dalam mengurangi beban pada sumber daya CPU dan memori selama serangan DDoS. Mitigasi yang diterapkan membantu memulihkan performa jaringan, mengurangi kemacetan, dan memastikan operasi jaringan tetap

stabil dan efisien. Meskipun ada beberapa peningkatan dalam penggunaan CPU oleh proses tertentu selama serangan ICMP *flood*, namun proses mitigasi mampu mengurangi dampak negatif serangan dan memperbaiki kondisi jaringan.

#### 4.4. Hasil dan Analisis Pembuatan *Dataset*

Pembuatan *dataset* merupakan tahap krusial dalam penelitian ini, yang bertujuan untuk mengembangkan model yang dapat dengan akurat mendeteksi dan memitigasi serangan DDoS dalam jaringan SDN. *Dataset* dikumpulkan dengan menggunakan metode simulasi, di mana perintah *ping* digunakan untuk menghasilkan lalu lintas jaringan normal dan *Hping3* digunakan untuk menghasilkan lalu lintas serangan DDoS, yang merefleksikan karakteristik lalu lintas jaringan yang beragam. Dibawah ini merupakan contoh *dataset* yang di hasilkan pada penelitian ini.

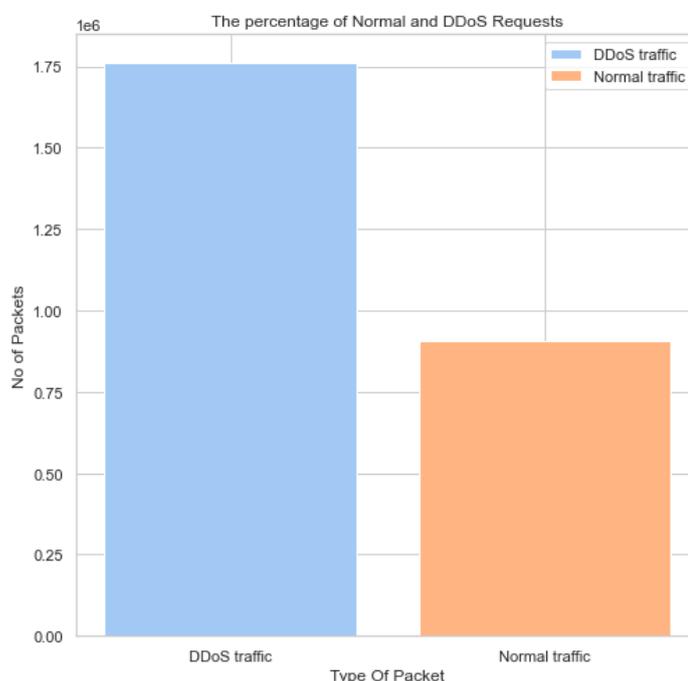


	timestamp	datapath_id	flow_id	ip_src	tp_src	ip_dst	tp_dst	ip
001	1590009949.044784	4	10.0.0.1010.0.0.1601	10.0.0.1	0	10.0.0.16	0	1
002	1590009949.044784	4	10.0.0.11010.0.0.601	10.0.0.11	0	10.0.0.6	0	1
003	1590009949.044784	4	10.0.0.14010.0.0.301	10.0.0.14	0	10.0.0.3	0	1
004	1590009949.044784	4	10.0.0.16010.0.0.101	10.0.0.16	0	10.0.0.1	0	1
005	1590009949.044784	4	10.0.0.3010.0.0.1401	10.0.0.3	0	10.0.0.14	0	1
006	1590009949.044784	4	10.0.0.6010.0.0.1101	10.0.0.6	0	10.0.0.11	0	1
007	1590009949.045698	2	10.0.0.1010.0.0.1601	10.0.0.1	0	10.0.0.16	0	1
008	1590009949.045698	2	10.0.0.11010.0.0.601	10.0.0.11	0	10.0.0.6	0	1
009	1590009949.045698	2	10.0.0.14010.0.0.301	10.0.0.14	0	10.0.0.3	0	1
010	1590009949.045698	2	10.0.0.16010.0.0.101	10.0.0.16	0	10.0.0.1	0	1
011	1590009949.045698	2	10.0.0.3010.0.0.1401	10.0.0.3	0	10.0.0.14	0	1
012	1590009949.045698	2	10.0.0.6010.0.0.1101	10.0.0.6	0	10.0.0.11	0	1
013	1590009949.046384	1	10.0.0.1010.0.0.1601	10.0.0.1	0	10.0.0.16	0	1
014	1590009949.046384	1	10.0.0.14010.0.0.301	10.0.0.14	0	10.0.0.3	0	1
015	1590009949.046384	1	10.0.0.16010.0.0.101	10.0.0.16	0	10.0.0.1	0	1

Gambar 4.15 Hasil *dataset* berdasarkan ekstraksi jaringan SDN

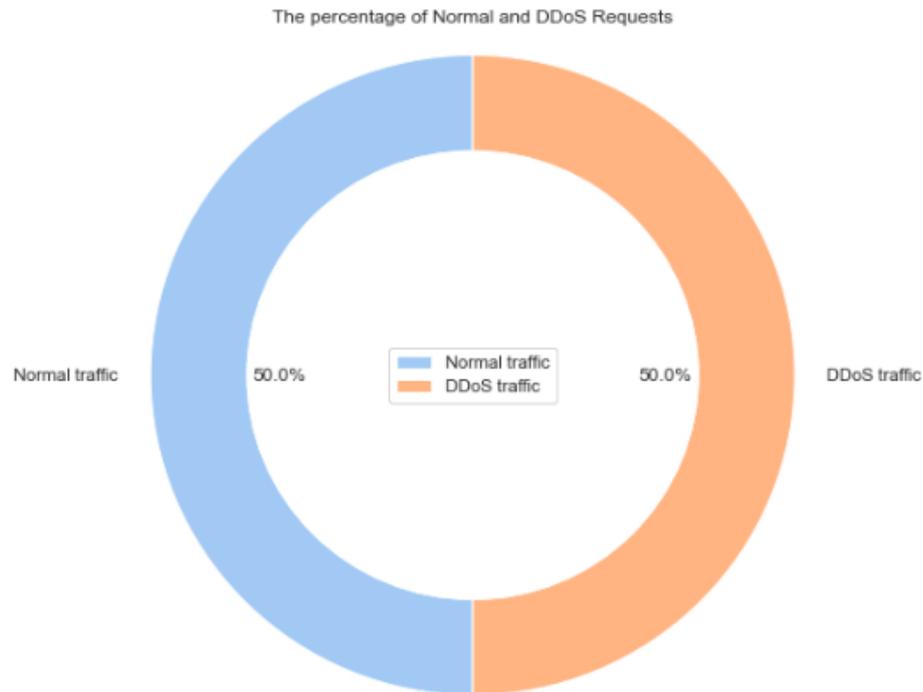
Berdasarkan gambar 4.15, *dataset* yang terkumpul terdiri dari dua kelas utama: lalu lintas normal dan lalu lintas serangan. Setiap kelas memiliki karakteristik tertentu yang tercermin dalam fitur-fitur *dataset*, seperti frekuensi paket, ukuran paket, dan interval waktu antar paket. Analisis awal *dataset*

menunjukkan bahwa ada perbedaan statistik yang signifikan antara dua kelas tersebut, yang memberikan dasar yang baik untuk pembelajaran model.



Gambar 4.16 Distribusi jumlah paket lalu lintas dalam *dataset*

Gambar 4.16 memberikan visualisasi yang jelas tentang komposisi *dataset* yang dikumpulkan sebelum proses *pre-processing*. Grafik tersebut mengilustrasikan perbandingan antara jumlah paket yang dikategorikan sebagai lalu lintas DDoS, yang mencapai 1,760,670 paket, dengan lalu lintas normal, yang berjumlah 906,853 paket. Selanjutnya merupakan tahap *pre-processing* yang bertujuan untuk menormalkan distribusi paket antara dua kelas sehingga model dapat belajar dengan representasi yang lebih seimbang dari kedua kondisi lalu lintas. Hal ini penting karena dengan menyelaraskan skala data, model MLP dapat memperoleh pemahaman yang lebih baik tentang variasi nilai yang ada dalam *dataset*. Selain itu, penghilangan nilai *NAN* juga dilakukan untuk mengurangi distorsi yang mungkin terjadi dalam data. Terakhir, teknik pemisahan data latih dan uji disusun untuk validasi model. Langkah ini memungkinkan model untuk diuji dengan data yang tidak pernah dilihat sebelumnya sehingga dapat mengevaluasi keefektifan dan keakuratan model dalam mendeteksi serangan DDoS. Berikut ini adalah visualisasi dari data hasil *pre-processing*.



Gambar 4.17 Hasil *pre-processing Dataset*

Gambar 4.17 menunjukkan hasil dari proses *pre-processing dataset*, di mana jumlah data antara lalu lintas normal dan DDoS telah diseimbangkan menjadi 50% untuk masing-masing kategori. Kesetaraan ini penting untuk memastikan bahwa model pembelajaran mesin, seperti MLP yang akan digunakan dalam penelitian ini, tidak bias terhadap satu kelas dan dapat belajar dengan akurat dari kedua jenis lalu lintas. Kesetaraan distribusi ini mendukung pembelajaran yang lebih efektif dan penilaian yang lebih adil terhadap kinerja model dalam mendeteksi serangan DDoS.

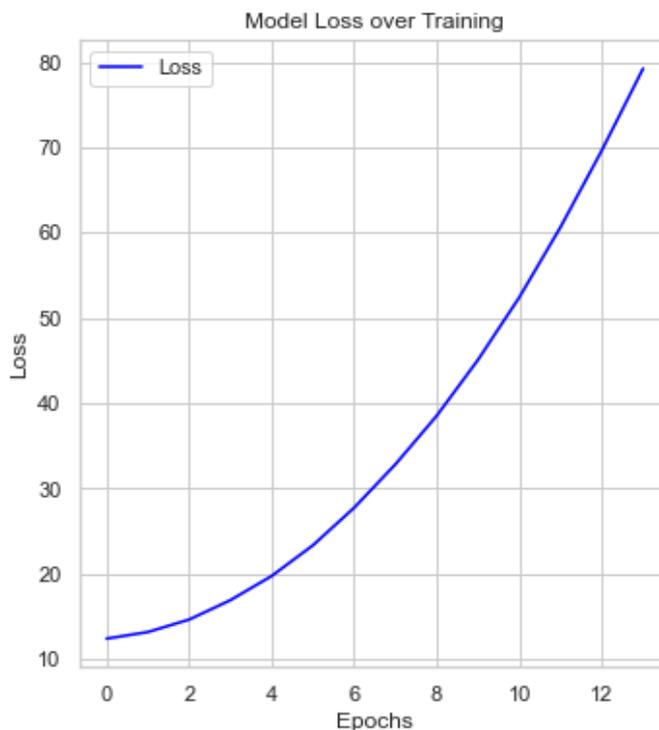
#### 4.5. Evaluasi Model MLP

Evaluasi merupakan aspek penting yang menentukan efektivitas model dalam mengklasifikasikan lalu lintas jaringan sebagai normal atau serangan DDoS. Dengan menggunakan metrik kinerja seperti akurasi, presisi, *recall*, dan *F1-score*, serta analisis kurva ROC. Analisis ini bertujuan untuk memberikan gambaran mengenai kinerja model, serta mengidentifikasi kelebihan serta kelemahan dari model. Dibawah ini merupakan grafik akurasi yang merupakan salah satu evaluasi model MLP.



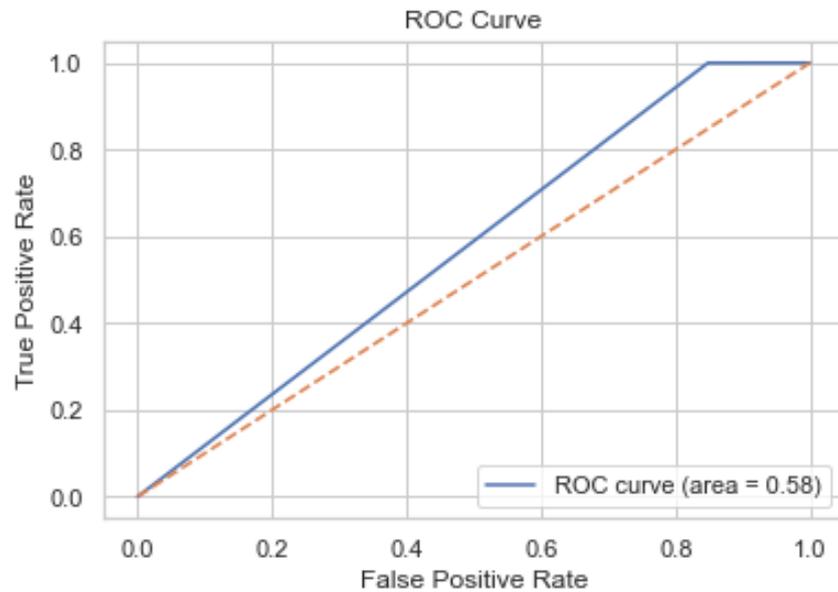
Gambar 4.18 Akurasi model MLP

Gambar 4.18 memperlihatkan tren akurasi model MLP dengan penggunaan teknik *early stopping* selama proses pelatihan. Grafik menunjukkan fluktuasi signifikan dalam akurasi, dengan beberapa puncak yang menembus di atas 70% dan kemudian menurun kembali. Selain itu, terlihat bahwa terdapat variasi yang cukup besar dalam akurasi pada setiap *epoch* yang menunjukkan adanya variasi yang signifikan dalam performa model. Berdasarkan konfigurasi model yang diberikan, di mana *early stopping* diaktifkan untuk menghindari *overfitting*, grafik menunjukkan bahwa pelatihan model berhenti secara otomatis sebelum semua *epoch* terlaksana, hal ini tercermin dari perubahan tajam pada akurasi yang terjadi pada *epoch* terakhir. Penerapan *early stopping* dengan 10% data sebagai validasi menjamin bahwa model tidak melanjutkan pelatihan yang berlebihan, yang dapat dilihat dari stabilisasi akurasi sebelum terjadinya peningkatan tajam tersebut. Hal ini menunjukkan bahwa model telah mencapai kemampuan prediksi yang baik pada data pelatihan dan data validasi. Dengan kata lain, model tidak terlalu fokus pada detail-detail kecil yang hanya ada pada data pelatihan, tetapi mampu mengidentifikasi pola-pola yang lebih umum dan dapat digeneralisasi pada data baru. Selain dari akurasi model, terdapat juga nilai dari kerugian model atau bisa disebut dengan *model loss*. Berikut ini merupakan visualisasi dari nilai kerugian model.



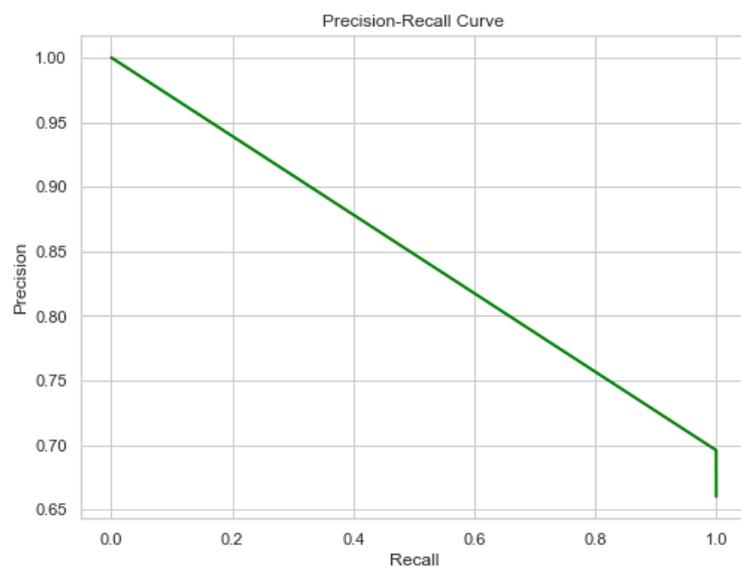
Gambar 4.19 Grafik *loss* model MLP

Gambar 4.19 menampilkan grafik kerugian model selama proses pelatihan dari model MLP. Grafik ini menggambarkan peningkatan kerugian yang terjadi seiring bertambahnya *epoch*. Peningkatan ini menunjukkan adanya isu pada proses pelatihan yang perlu ditangani. Kenaikan kerugian yang terus-menerus seperti ini menunjukkan bahwa model mungkin mengalami kesulitan dalam menyesuaikan bobot untuk mengurangi kesalahan prediksi. Dalam hal ini, penyesuaian bobot yang tidak efektif dapat mengarah pada peningkatan kerugian yang signifikan selama proses pelatihan. Dalam konteks ini, peningkatan kerugian yang terus-menerus memicu aktivasi *early stopping* untuk mencegah pembelajaran yang berlebihan dan tidak efektif. Tujuannya adalah untuk mencegah model dari *overfitting*, di mana model terlalu memfokuskan pada detail kecil yang hanya ada pada data pelatihan tetapi tidak dapat digeneralisasi dengan baik pada data baru. Dengan demikian, model MLP dapat mencapai hasil yang optimal dan dapat digunakan untuk deteksi serangan DDOS dengan akurasi yang tinggi. Dibawah ini merupakan grafik dari ROC yang akan menunjukkan kinerja model dengan cara membandingkan nilai dari *true positif* dan nilai dari *false positive*.



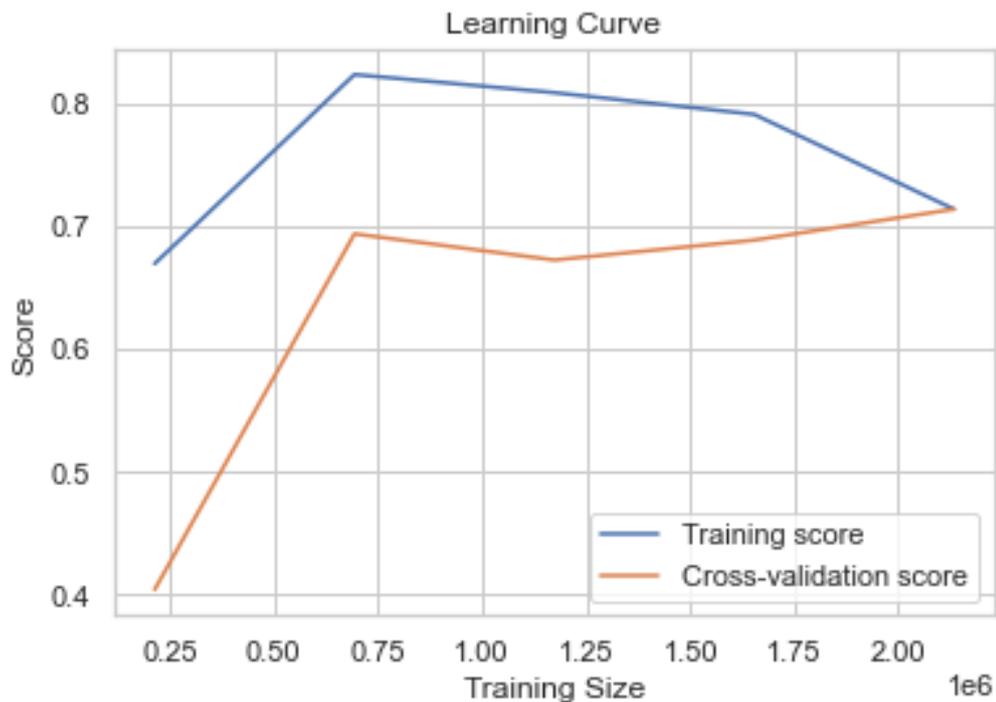
Gambar 4.20 Grafik *Receiver Operating Characteristic* (ROC)

Gambar 4.20 menunjukkan kurva *Receiver Operating Characteristic* (ROC) untuk model MLP yang telah dilatih. Dari grafik tersebut, dapat kita lihat bahwa area di bawah kurva AUC memiliki nilai sebesar 0.58. Nilai tersebut menunjukkan bahwa model ini tidak memiliki kemampuan diskriminasi yang baik dalam membedakan antara kelas positif dan negatif. Untuk menjadi model yang ideal, kurva ROC seharusnya mendekati sudut kiri atas dan memiliki AUC yang mendekati 1.



Gambar 4.21 Grafik presisi-*recall*

Gambar 4.21 yang ditampilkan di bawah ini menunjukkan grafik Precision-Recall yang digunakan untuk mengevaluasi performa model MLP dalam konteks klasifikasi. Grafik ini memberikan gambaran tentang hubungan antara presisi dan *recall*. Presisi merupakan nilai keakuratan model dalam memprediksi nilai positif, sedangkan *recall* adalah nilai yang mengukur keberhasilan model dalam mendeteksi seberapa banyak kasus positif yang berhasil diidentifikasi oleh model. Dari analisis grafik tersebut, dapat terlihat bahwa presisi cenderung menurun seiring dengan peningkatan *recall*. Hal ini mengindikasikan bahwa ketika model berusaha untuk mengidentifikasi lebih banyak kasus positif atau *recall* yang tinggi, kemampuan model untuk menjaga keakuratan prediksinya atau presisi cenderung menurun. Terdapat penurunan yang signifikan pada nilai presisi saat *recall* mendekati 1.0, menunjukkan bahwa model tersebut mungkin mengklasifikasikan banyak kasus negatif sebagai positif dalam rangka mencapai *recall* yang tinggi.



Gambar 4.22 Kurva pembelajaran model

Gambar 4.22 menyajikan grafik learning curve yang menggambarkan perbandingan antara skor pelatihan dan skor validasi silang model MLP terhadap berbagai ukuran set pelatihan. Berdasarkan grafik di atas skor pelatihan meningkat seiring bertambahnya data, menunjukkan peningkatan performa model pada data

yang dikenal. Namun, setelah melewati titik tertentu, skor pelatihan menunjukkan penurunan, yang bisa mengindikasikan *overfitting*. Di sisi lain, skor validasi silang meningkat dan setelah mencapai puncak, menurun sedikit sebelum stabil, menandakan model mulai menggeneralisasi dengan lebih baik terhadap data yang tidak dikenal. Stabilitas skor validasi silang pada akhirnya menunjukkan bahwa model telah mempelajari pola yang relevan dari *dataset*.

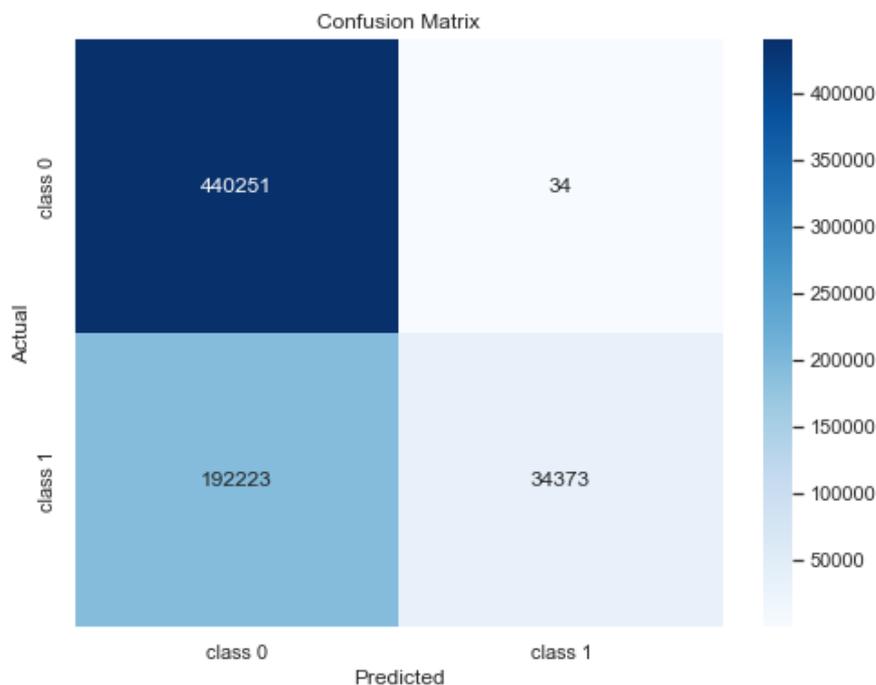
#### 4.6. Efektivitas Pelatihan Model

Pada bagian ini, akan dibahas mengenai efektivitas pelatihan model MLP dalam penelitian ini. Evaluasi dan analisis yang dilakukan bertujuan untuk memahami sejauh mana model MLP dapat mengklasifikasikan lalu lintas jaringan sebagai normal atau serangan DDoS. Berikut adalah laporan hasil klasifikasi, *confusion matrix*, dan dataframe hasil prediksi.

	precision	recall	f1-score	support
0	1.00	0.15	0.26	226596
1	0.70	1.00	0.82	440285
accuracy			0.71	666881
macro avg	0.85	0.58	0.54	666881
weighted avg	0.80	0.71	0.63	666881

Gambar 4.23 Laporan hasil klasifikasi

Gambar 4.23 menampilkan laporan klasifikasi yang mengukur efektivitas model MLP yang telah dilatih. Model ini menunjukkan presisi sempurna untuk kelas 0 (non-DDoS) tetapi dengan *recall* yang rendah, mengindikasikan bahwa sementara semua prediksi positif untuk kelas ini benar, banyak kasus positif sebenarnya yang terlewat. Sebaliknya, untuk kelas 1 (DDoS), model memiliki *recall* sempurna namun presisi yang lebih rendah, menandakan bahwa meskipun semua kasus positif sebenarnya teridentifikasi, terdapat banyak kasus negatif yang salah diklasifikasikan sebagai positif. Akurasi keseluruhan model adalah 71%. Akurasi ini dianggap baik karena mampu mencapai tujuan yaitu mendeteksi DDoS dan membedakannya dengan lalu lintas jaringan normal. Dengan akurasi sebesar 71%, sistem mampu mencapai tujuan yaitu mendeteksi dan memitigasi serangan DDoS serta membedakannya dari lalu lintas normal.



Gambar 4.24 *Confusion matrix* model

Gambar 4.24 menampilkan matriks kebingungan (*confusion matrix*) untuk model MLP yang telah diuji. Dari matriks ini, dapat dilihat beberapa hal penting. Pertama, model MLP berhasil mengklasifikasikan 440,251 kasus sebagai kelas negatif (kelas 0) dengan benar. Ini menunjukkan kemampuan model dalam mengenali kasus negatif. Namun, ada hal yang perlu diperhatikan, yaitu model juga salah mengklasifikasikan 192,223 kasus positif (kelas 1) sebagai negatif. Ini menandakan bahwa ada ruang untuk peningkatan kualitas klasifikasi model ini.

Selanjutnya, jika dilihat lebih detail, terdapat 34 kasus negatif yang salah diklasifikasikan sebagai positif. Meskipun jumlahnya tidak besar, namun hal ini tetap perlu diperbaiki agar klasifikasi lebih akurat. Namun, yang menarik adalah bahwa 34,373 kasus positif berhasil diklasifikasikan dengan benar. Ini menunjukkan keunggulan model dalam mengidentifikasi kasus positif.

Namun, perlu diakui adanya kecenderungan tinggi dari model untuk mengklasifikasikan kasus sebagai negatif. Hal ini bisa disebabkan oleh ketidakseimbangan kelas dalam data pelatihan atau mungkin ada bias dalam model itu sendiri. Oleh karena itu, untuk meningkatkan kualitas klasifikasi, perlu dilakukan penanganan terhadap ketidakseimbangan kelas dan evaluasi lebih lanjut terhadap model.

Sample Data:

	Actual	Predicted
0	1	1
1	0	0
2	1	1
3	0	1
4	1	1

Model Accuracy: 71.17%

Gambar 4.25 Dataframe hasil prediksi

Gambar 4.25 menampilkan sebuah dataframe yang mengilustrasikan hasil prediksi dari model *Multilayer Perceptron* (MLP) dengan menggunakan contoh data sampel yang telah disediakan. DataFrame ini memberikan gambaran tentang performa model dalam melakukan prediksi. Kolom *Actual* menunjukkan label sebenarnya dari data, sedangkan kolom *Predicted* menunjukkan label yang diprediksi oleh model. Kesesuaian antara kedua kolom ini menunjukkan sejauh mana prediksi model dapat menghasilkan hasil yang akurat.

Dalam sampel yang ditampilkan, terdapat beberapa kesalahan prediksi yang dapat dilihat dari perbedaan nilai antara kolom *Actual* dan *Predicted*. Meskipun demikian, secara keseluruhan, model MLP mencapai tingkat akurasi sebesar 71.17%. Ini berarti bahwa model berhasil memprediksi dengan benar sekitar 71.17% dari data yang diberikan. Namun, berdasarkan pengujian yang merujuk pada Tabel 4.1, yaitu hasil deteksi sistem, menunjukkan akurasi saat pengujian mencapai 100%. Oleh karena itu, akurasi ini dianggap mampu mendeteksi dan memitigasi serangan DDoS secara efektif. Dengan adanya gambaran ini, dapat disimpulkan bahwa model MLP telah memberikan hasil prediksi yang cukup baik, meskipun masih terdapat ruang untuk perbaikan.