Analisis Jaringan Taktis Pada *Software Defined Network* (SDN) Dengan Menggunakan *Intrusion Detection And Prevention System* (IDPS) Untuk Mengevaluasi Efektivitas Keamanan Pada Jaringan

Aprilia Intan Prasetya
Fakultas Informatika
Direktorat Kampus Universitas Telkom Purwokerto
Purwokerto, Indonesia
apriliaintanprasetya@student.telkomuniversity.ac.id

Wahyu Adi Prabowo
Fakultas Informatika
Direktorat Kampus Universitas Telkom Purwokerto
Purwokerto, Indonesia
wahyup@telkomuniversity.ac.id

Alon Jala Tirta Segara
Fakultas Informatika
Direktorat Kampus Universitas Telkom Purwokerto
Purwokerto, Indonesia
alonhs@telkomuniversity.ac.id

***Abstrak* — Serangan *Distributed Denial of Service* (*DDoS*) *SYN Flood* merupakan ancaman serius terhadap jaringan berbasis *Software Defined Networking* (*SDN*). *Topologi* eksperimen melibatkan *SDN controller*, *Open vSwitch*, *web server*, *router PFSense*, serta dua jaringan tambahan sebagai *client* dan *attacker*. Sistem keamanan dirancang dengan mengintegrasikan *IDPS*, serta pemfilteran trafik menggunakan *iptables* dan *ipset*. Evaluasi performa dilakukan berdasarkan parameter *Quality of Service* (*QoS*), yaitu *throughput*, *packet loss*, *delay*, dan *jitter*. Hasil menunjukkan bahwa *IDPS* menurunkan *throughput* saat serangan dari 1095,10 bit/s menjadi 455,45 bit/s, serta pada kondisi minimal dari 3,33 bit/s menjadi 1,34 bit/s. *Packet loss* tercatat 0% pada tiga skenario, kecuali pada *Non IDPS Flooding* sebesar 0,00075295%. *Delay* tertinggi terjadi pada *IDPS* Minimal yaitu 381,72 ms dan terendah pada *Non IDPS* Minimal 145,35 ms. *Jitter* tertinggi terdapat pada *IDPS Flooding* 1084,73 ms, diikuti *IDPS* Minimal 379,21 ms. Temuan ini menunjukkan efektivitas *IDPS* dalam mitigasi serangan, meski berdampak pada *delay* dan *jitter*. Sistem terbukti menjaga kestabilan jaringan dan mempertahankan *QoS* selama serangan berlangsung.*

Kata kunci*— *DDoS, SYN Flood, SDN, IDPS, QoS.

I. PENDAHULUAN

Jaringan komputer telah menjadi bagian integral dalam kehidupan manusia. Namun, semakin besar ketergantungan kita pada jaringan, semakin besar pula risiko keamanan yang dihadapi di era digital ini. Berdasarkan data *survey* yang dilakukan oleh *We Are Social* pada Januari 2024, terdapat 5,35 miliar pengguna internet di seluruh dunia, yang merupakan 66,2% dari populasi global. Dari jumlah tersebut, 5,35 miliar, atau 66,2% dari populasi dunia, adalah pengguna media sosial [1]. Sedangkan menurut data yang diumumkan oleh APJII, jumlah pengguna *internet* Indonesia tahun 2024 mencapai 221.563.479 jiwa dari total populasi 278.696.200 jiwa penduduk Indonesia tahun 2023. Dari hasil survei penetrasi internet Indonesia 2024 yang dirilis APJII, maka tingkat penetrasi *internet* Indonesia menyentuh angka 79,5%. Dibandingkan dengan periode sebelumnya, maka ada peningkatan 1,4% [2].

Dengan bertambahnya jumlah pengguna *internet* di dunia, maka semakin meningkat pula tindakan serangan siber (*cybercrime*) atau peretasan yang di lakukan pada suatu jaringan internet. Menurut data yang diumumkan oleh Menteri Komunikasi dan Informasi, Indonesia berada di urutan kedua untuk *cybercrime* dengan persentase terbanyak yaitu peretasan dalam dunia maya dan sektor perbankan[3]. Serangan siber atau *cybercrime* sendiri merupakan upaya yang disengaja untuk mencuri, mengekspos, mengubah, melumpuhkan, atau menghancurkan data, aplikasi, atau aset lainnya melalui akses tidak sah ke jaringan, sistem komputer, atau perangkat digital.

Data statistik dari Badan Siber dan Sandi Negara (BSSN) mencatat bahwa telah terjadi 370,02 juta serangan siber terhadap Indonesia pada tahun 2022. Dibandingkan dengan tahun sebelumnya terjadi 266,74 juta serangan siber, jumlah ini meningkat sebesar 38,72% [4]. Pada kuartal pertama 2025, tercatat lebih dari 3,2 juta ancaman siber terdeteksi oleh *Kaspersky Security Network* (KSN) yang memengaruhi sekitar 15,5 % pengguna di Indonesia[5].

Melihat perkembangan teknologi informasi, ancaman kejahatan siber saat ini sangat penting Ancaman baru seperti *Cyber War* atau perang siber. Beberapa contoh perang atau serangan siber yang terjadi adalah *Email propagation* of malicious *code*, *Wide-scale Trojan distribution, Distributed attack tools, Distributed Denial of service (DDoS) attacks, Targeting of specific users, Antiforensic techniques, Wide-scale use of worms, dan Sophisticated command and control attacks* [6].

Salah satu bentuk *DDoS* yang patut diwaspadai adalah *SYN flood*, yakni serangan berbasis protokol *TCP* yang mengeksploitasi kelemahan dalam proses *three-way handshake*. Dalam serangan ini, penyerang membanjiri *server* dengan paket *SYN* tanpa menyelesaikan koneksi, sehingga menyebabkan server kehabisan sumber daya karena menyimpan banyak koneksi setengah terbuka (*half-open connections*). Dampak dari serangan ini bisa sangat parah, terutama terhadap sistem yang memiliki kontrol terpusat seperti pada arsitektur *Software Defined Networking* (*SDN*).

Arsitektur *SDN* mencakup pengontrol jaringan terpusat dengan pengontrol jaringan global dari jaringan dan *Application Programming Interface* (API) untuk mengembangkan aplikasi jaringan. *SDN* beroperasi sebagai "otak" dari sistem, yang mengontrol sakelar jaringan dan memelihara semua fungsi jaringan melalui pemantauan ekstensif dan administrasi [7]. Serangan terhadap *controller SDN* dapat berakibat fatal bagi seluruh jaringan. *Controller* memegang kendali penuh atas konfigurasi dan aliran trafik. Serangan terhadap *controller* dapat berakibat fatal, memungkinkan penyerang untuk memanipulasi konfigurasi jaringan, mengalihkan trafik ke server jahat, memblokir akses ke sumber daya penting, atau bahkan meluncurkan serangan lebih lanjut ke perangkat lain di jaringan.

Untuk menghadapi tantangan tersebut, perlu diimplementasikan sistem keamanan yang andal, salah satunya adalah *Intrusion Detection and Prevention System* (*IDPS*). *IDPS* merupakan teknologi yang berfungsi untuk mendeteksi, memantau, dan mencegah aktivitas mencurigakan dalam jaringan. Penelitian ini memiliki fokus utama pada pengukuran efektivitas jaringan *SDN* ketika menerima serangan *flooding* secara bersamaan dalam skala yang berbeda, serta mengevaluasi peran *IDPS* dalam menjaga kestabilan jaringan. Pengujian ini dilakukan untuk mengetahui seberapa besar dampak perbedaan intensitas serangan terhadap parameter kualitas layanan jaringan (*Quality of Service/QoS*) seperti *throughput*, *jitter*, *delay*, dan *packet* loss.

II. KAJIAN TEORI

A. *Software Defined Network* (SDN)

Software Defined Networking (SDN) adalah teknik yang memanfaatkan perangkat lunak manajemen khusus untuk meningkatkan kecerdasan dan fleksibilitas jaringan area luas. *SDN* dapat membantu jaringan area luas menangani lalu lintas jaringan dengan protokol tertentu dan menyedihkan perangkat keras jaringan *[8]*.

B. *Intrusion Detection and Prevention System* (IDPS)

Intrusion Detection and Prevention System (IDPS) merupakan *hardware* atau *software* yang dirancang untuk mendeteksi dan mencegah intrusi jaringan atau sistem komputer [9].

C. *Distributed Denial of Servie* (DDoS)

DDoS adalah tindakan pengiriman paket dalam jumlah besar ke jaringan dengan tujuan membanjiri jaringan dengan data sehingga pengguna yang berhak tidak dapat mengakses *host* [10] [11].

D. SYN Flood

SYN Flood merupakan salah satu jenis serangan *Denial of Service* (*DoS*) yang secara spesifik mengeksploitasi kelemahan dalam proses *three-way handshake* pada protokol *TCP* [12].

E. Quality of Service (QoS)

Quality of Service (QoS) atau Kualitas Layanan adalah sebuah metode digunakan untuk mengukur kualitas jaringan dan menentukan tingkat layanan yang disediakannya [13]. Ada 4 kategori penilaian pada QoS untuk menilai sebuah layanan jaringan antara lain:

1. Throughput

Throughput merupakan kecepatan (*rate*) transfer data efektif, yang diukur dalam *bps* [13]. Penilaian Throughput dapat dilakukan dengan persamaan sebagai berikut [13]:

$$Throughput = \frac{\text{Paket Data diterima}}{\text{Lama Pengamatan}} \quad (2.1)$$

2. Packet Loss

Packet loss merupakan suatu parameter yang menunjukkan jumlah paket yang hilang total yang dapat terjadi karena gangguan dan benturan jaringan [13]. Penilaian packet loss dapat dilakukan dengan persamaan sebagai berikut [13]:

$$Packet\ Loss = \frac{\text{Paket Data dikirim} - \text{Paket Data diterima}}{\text{Paket Data dikirim}} \text{ x } 100\% \quad (2.2)$$

3. Delay

Delay merupakan jumlah waktu yang dibutuhkan data untuk menghitung jarak dari awal ke tujuan[13]. Penilaian Throughput dapat dilakukan dengan persamaan sebagai berikut [13]:

$$Delay = \frac{\text{total delay}}{\text{total paket yang diterima}} \quad (2.3)$$

4. Jitter

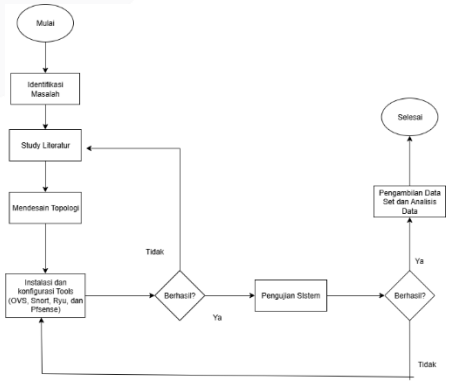
Jitter merupakan kemacetan dapat menyebabkan paket datang terlambat atau tidak berurutan pada jaringan, yang dapat menyebabkan distorsi atau kesenjangan dalam transmisi [13]. Penilaian jitter dapat dilakukan dengan persamaan berikut [13]:

$$\text{Jitter} = \frac{\text{total variasi delay}}{\text{total paket yang diterima}} \quad (2.4)$$

III. METODE

Penelitian ini bertujuan untuk mengetahui efektifivitas dari integrasi antara IDPS yang menggunakan Snort dan SDN yang menggunakan Ryu dalam menghadapi serangan DDoS SYN Flood dengan mode serangan yang berbeda. Kemudian akan dianalisis dengan menggunakan parameter QoS yang meliputi throughput, packet loss, delay dan jitter.

A. Alur Penelitian



GAMBAR 1
(DIAGRAM ALIR PENELITIAN)

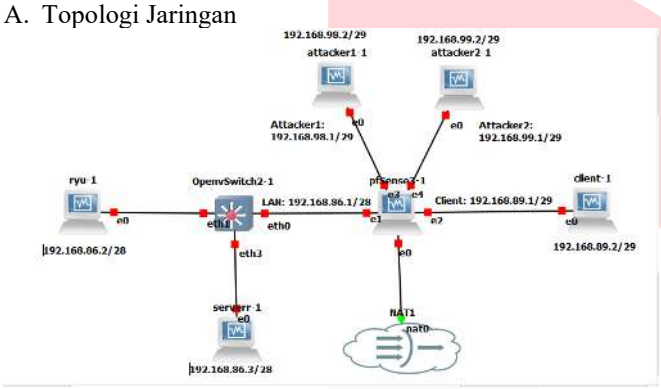
Penelitian ini diawali dengan identifikasi masalah untuk menentukan fokus dan ruang lingkup penelitian. Selanjutnya

dilakukan studi literatur terkait SDN, IDPS serta mengenai keamanan jaringan guna memperoleh landasan teori, konsep, dan referensi teknologi yang relevan. Berdasarkan hasil studi, dilakukan perancangan topologi yang mencakup arsitektur jaringan dan komponen pendukung sistem.

Tahap berikutnya adalah instalasi dan konfigurasi tools seperti Open vSwitch (OVS), Snort, Ryu, dan PfSense, dilanjutkan dengan pengujian sistem untuk memverifikasi fungsi dan kinerja. Hasil pengujian dievaluasi untuk menentukan apakah sudah sesuai dengan tujuan penelitian.

Tahap terakhir adalah pengambilan data dan analisis data. Data yang diperoleh dari pengujian dianalisis untuk menghasilkan temuan penelitian. Apabila semua tahapan telah terpenuhi dan sistem berfungsi optimal, penelitian dinyatakan selesai.

A. Topologi Jaringan



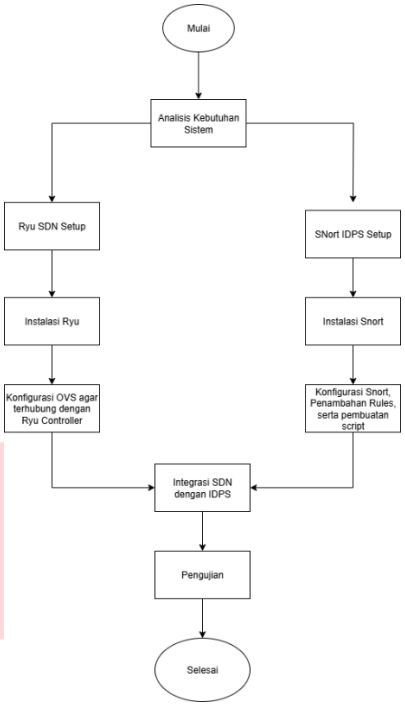
GAMBAR 2
(TOPOLOGI JARINGAN)

Gambar 2 merupakan topologi jaringan yang digunakan dalam penelitian ini. Topologi jaringan terdiri dari atas 3 jaringan yang berbeda dengan *IP Network* yang berbeda. Jaringan pertama yaitu ada *LAN* (*Local Address Network*), di jaringan *LAN* terdapat beberapa perangkat seperti *Switch OpenvSwitch* yang bertugas untuk menghubungkan perangkat di jaringan *LAN* dan juga sebagai *switch* virtual agar *SDN Ryu*, dan juga *Sever*. Selain itu *OVS* juga bertindak sebagai penghubung antara jaringan *LAN* dan juga *Router PfSense*.

Controller bertindak sebagai otak kanan jaringan yang dimana *controller* (*Ryu*) bertugas memantau jaringan. *Server* bertugas sebagai perangkat yang menyediakan informasi yang bisa client akses dan juga berperan menjadi *Snort* sebagai *IDPS* (*Intrusion Detection and Prevention System*) yang bertugas untuk mendeteksi dan juga memblokir serangan dari *Attacker* yang berusaha menyerang dengan membanjiri *request* kepada *Server*. *Router PfSense* bertugas untuk menghubungkan jaringan *LAN*, *Client* dan juga *Attacker*.

Jaringan *Client* terdiri dari 1 komputer yang terhubung dengan router *PfSense* untuk mengakses layanan yang ada pada server. Jaringan terakhir yaitu jaringan *Attacker* atau penyerang yang dimana terdapat 2 komputer yang bertugas untuk menyerang *Server* dengan cara membanjiri *request* atau menggunakan serangan *DDoS SynFlood*.

B. Konfigurasi Sistem



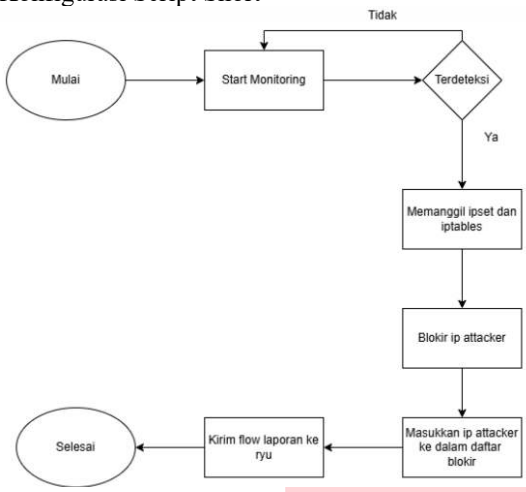
GAMBAR 3
(KONFIGURASI SISTEM)

Gambar 3 merupakan gambar Konfigurasi Sistem SDN dengan IDPS. Alur diawali pada tahap Mulai, kemudian langsung menuju Analisis Kebutuhan Sistem. Setelah kebutuhan sistem jelas, alur bercabang menjadi dua jalur paralel. Jalur kiri adalah *Ryu* SDN *setup*, dimulai dari proses *instalasi* Ryu, kemudian dilanjutkan dengan konfigurasi *OVS* agar terhubung dengan *Ryu Controller*.

Sementara jalur kanan adalah *Snort IDPS setup*, dimulai dengan Instalasi *Snort*, setelah *Snort* terpasang, proses dilanjutkan Konfigurasi *Snort*, Penambahan *Rules*, serta Pembuatan *Script*. Di tahap ini, Snort disiapkan untuk mendeteksi serangan *SYN Flood*, dan script otomatis dibuat agar *Snort* dapat memanggil *ipset/iptables* sekaligus mengirim laporan ke *Ryu*.

Kedua jalur kemudian bergabung pada tahap Integrasi *SDN* dengan *IDPS*, yaitu menghubungkan *Snort* yang bertugas mendeteksi/memblokir dengan *Ryu Controller* yang mencatat laporan. Selanjutnya, dilakukan Pengujian, yaitu menjalankan skenario serangan *SYN Flood* dan memantau apakah *Snort* berhasil mendeteksi, memblokir *IP* penyerang, dan mengirim laporan ke *Ryu*. Setelah pengujian selesai, alur penelitian ditutup pada tahap selesai.

C. Konfigurasi Script Snort

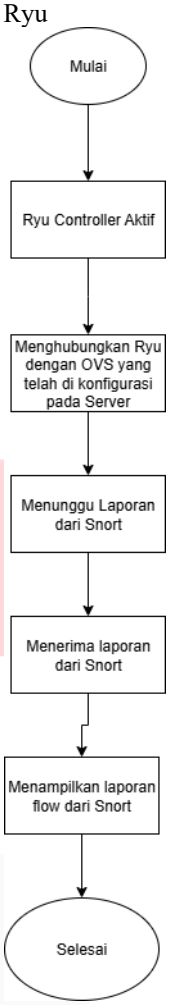


GAMBAR 4
(CARA KERJA SCRIPT SNORT)

Gambar 4 menggambarkan alur kerja *script Snort* untuk memantau trafik, mendeteksi serangan, dan secara otomatis melakukan tindakan mitigasi. Proses dimulai dengan *Snort Monitoring*, artinya *Snort* dalam kondisi aktif menjalankan rules. *Script* akan terus mengevaluasi apakah ada serangan yang terdeteksi. Pada Terdeteksi?, proses bercabang: jika Tidak, monitoring berulang terus menerus. Jika Ya, *script* akan segera mengeksekusi pada langkah eksekusi.

Langkah mengeksekusi pertama adalah fungsi memanggil *ipset* dan *iptables*, yaitu *script* akan *update* set *iptables* dan *ipset* untuk memblokir *IP* penyerang. Kemudian dilanjutkan memblokir *IP attacker*, yang berarti paket berikutnya dari *IP* itu akan di-*drop*. Selanjutnya, *IP* tersebut juga akan memasukkan *IP attacker* ke daftar blokir, sehingga saat *script* diulang. Tahap terakhir, *script* akan kirim *flow* laporan ke *Ryu*, yang memberitahukan kepada *controller* bahwa terdapat serangan pada *server* dan *IP* penyerang telah diblokir. Ini mempermudah *Ryu* untuk *logging* atau tindakan lanjutan. Proses lalu dinyatakan selesai.

D. Konfigurasi Script Ryu



GAMBAR 5
(CARA KERJA SCRIPT RYU)

Gambar 5 menggambarkan bagaimana *script controller Ryu* bekerja dalam keseluruhan sistem mitigasi serangan *SYN Flood*. Alur dimulai dengan *Ryu Controller* aktif, yaitu kondisi ketika *Ryu* sudah dijalankan pada *server, controller* dan siap menerima koneksi dari *Open vSwitch* (*OVS*). Langkah berikutnya adalah menghubungkan *Ryu* dengan *OVS* yang telah dikonfigurasi pada *server*.

Proses selanjutnya adalah menunggu laporan dari *Snort*, pada tahap ini *Ryu* berada dalam *state listening*, menunggu data yang dikirim oleh *script Snort*. Begitu ada aktivitas mencurigakan yang terdeteksi *Snort* dan *script Snort* melakukan pengiriman laporan, *Ryu* masuk ke tahap menerima laporan dari *Snort*. Setelah laporan diterima, *script* di *Ryu* akan menampilkan laporan *flow* dari *Snort*. Terakhir, proses dinyatakan Selesai.

IV. HASIL DAN PEMBAHASAN

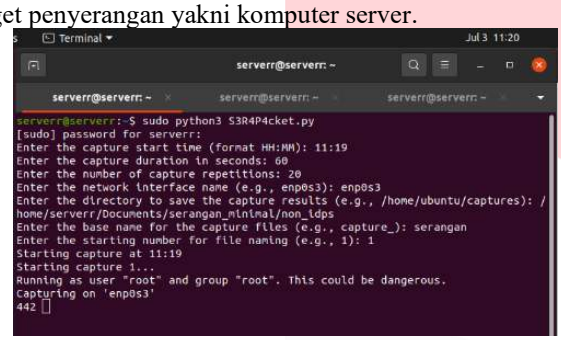
A. Skenario Penyerangan dan Pengujian Sistem

Pengujian yang dilakukan dalam penelitian yaitu dengan melakukan percobaan penyerangan yang dilakukan oleh dua komputer attacker dengan target komputer server. Selain itu, pengujian dilakukan dengan empat kondisi dalam waktu masing-masing 20 menit dan kemudian akan dianalisis dampaknya dengan menggunakan parameter QoS

1. Serangan SYN Flood Minimal Tanpa IDPS

GAMBAR 6
(PENGUJIAN SYN FLOOD MINIMAL TANPA IDPS)

Gambar 6 menunjukkan proses pengujian serangan DDoS SYN Flood minimal. Terdapat dua komputer attacker menghasilkan lalu lintas serangan secara terus-menerus ke target penyerangan yakni komputer server.



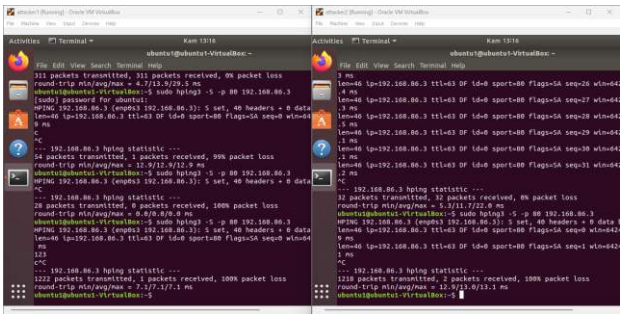
GAMBAR 7
(REKAM LALU LINTAS SAAT SERANGAN SYN FLOOD MINIMAL TANPA IDPS)

Pada gambar 7 menunjukkan pada *PC Server script python* S3R4P4cket.py sedang melakukan rekam lalu lintas jaringan ketika pengujian serangan dilakukan.

GAMBAR 8
(CAPTURE WIRESHARK SERANGAN SYN FLOOD MINIMAL TANPA IDPS)

Gambar 8 menunjukan hasil tangkapan paket dari *Wireshark* yang merekam lalu lintas jaringan selama simulasi serangan *SYN Flood* dilakukan. Terlihat ada serangkaian paket TCP dari alamat sumber 192.168.99.2 yang ditujukan ke alamat tujuan 192.168.86.3 pada *port* 80. Paket-paket ini berisi *flag* [*SYN*], yang menunjukkan permintaan awal untuk membuat koneksi *TCP*. Sebagai respon, *server* 192.168.86.3 membalas dengan paket [*SYN, ACK*], menandakan bahwa server bersedia melanjutkan negosiasi tiga langkah *handshake TCP*, tanpa IDPS serangan ini tidak terdeteksi maupun diblokir sehingga seluruh proses respons ditangani langsung oleh server.

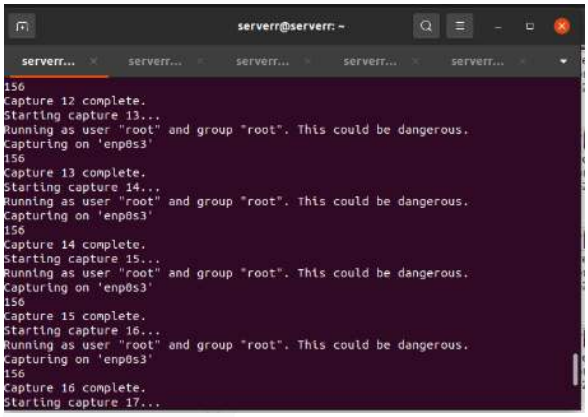
2. Serangan SYN Flood Minimal Dengan IDPS



GAMBAR 9
(PENGUJIAN SYN FLOOD MINIMAL DENGAN IDPS)

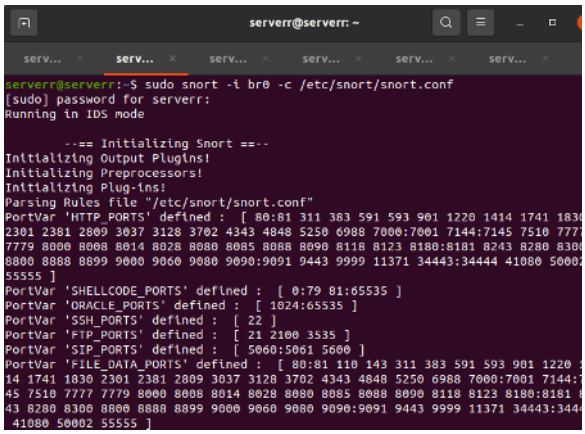
Gambar 9 menunjukkan aktivitas serangan *SYN Flood* minimal menggunakan *hping3* ke server target 192.168.86.3 dalam keadaan *IDPS* sudah aktif. Di sisi kiri, tampak tiga kali eksekusi serangan dengan jumlah paket yang relatif rendah dengan hasil packet loss mendekati 100%. Hal ini menjadi indikasi bahwa *IDPS* berhasil mendeteksi dan memblokir sebagian besar koneksi *SYN* palsu.

Panel terminal kanan menampilkan serangan yang sama dari mesin attacker kedua. Statistik menunjukkan 120 paket *SYN* dikirim, hanya 2 yang mendapat balasan, dengan packet loss sekitar 98%. Ini menegaskan efektivitas *IDPS* dalam memfilter trafik mencurigakan meskipun berasal dari sumber berbeda.



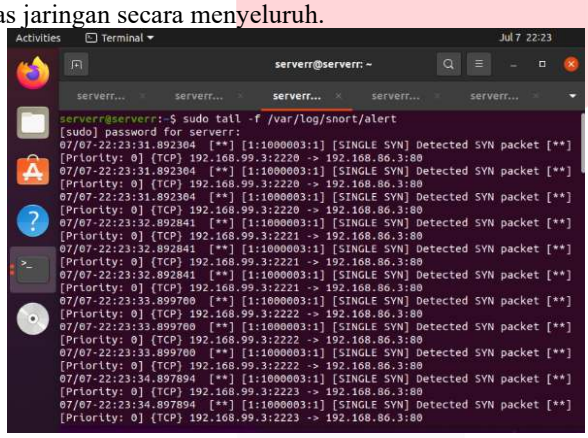
GAMBAR 10
(REKAM LALU LINTAS SAAT SERANGAN SYN FLOOD MINIMAL DENGAN IDPS)

Pada gambar 10 menunjukkan pada *PC Server script python S3R4P4cket*.py sedang melakukan rekam lalu lintas jaringan ketika pengujian serangan dilakukan. Script ini mencatat semua paket yang melintasi interface *enp0s3* dan paket tersebut akan disimpan dalam format *.pcap* agar dapat di analisis. Hasil rekam lalu lintas yang terekam jauh lebih sedikit jika dibandingkan dengan rekam lalu lintas tanpa *IDPS* yang aktif. Ini membuktikan bahwa *IDPS* sudah secara aktif *drop*, *blok* dan filtering paket-paket yang masuk.



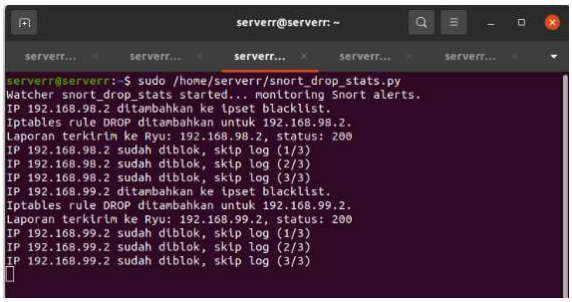
GAMBAR 11
(TERMINAL SAAT SNORT DIAKTIFKAN)

Gambar 11 menunjukkan proses inisialisasi *Snort* yang dijalankan pada mode *IDS* (*Intrusion Detection System*) di server. Proses inisialisasi ini merupakan tahap awal yang penting karena memastikan seluruh komponen pendukung *Snort* sudah dimuat dan siap untuk melakukan pemantauan lalu lintas jaringan secara menyeluruh.



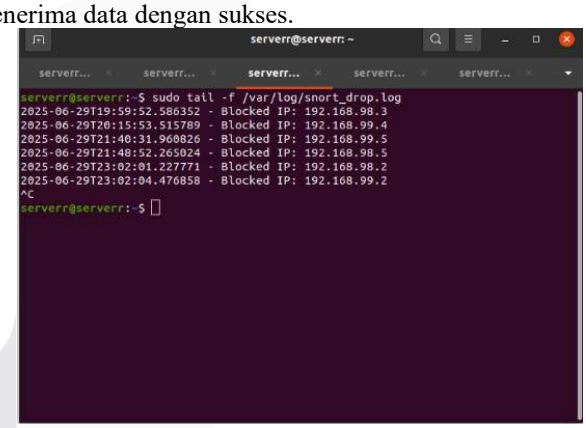
GAMBAR 12
(LOG ALERT SAAT TERJADI SERANGAN)

Gambar 12 menunjukkan tampilan keluaran log hasil *monitoring file /var/log/snort/alert* yang merekam aktivitas deteksi serangan *SYN Flood*. Hasilnya terlihat serangkaian notifikasi yang secara konsisten mendeteksi paket *SYN* yang mencurigakan. Setiap baris log mencantumkan waktu deteksi (07/07-22:23:13 dst.), *priority* (*Priority*: 0, berarti tingkat urgensi paling tinggi), protokol (*TCP*), alamat sumber (192.168.89.3), alamat tujuan (192.168.86.3), dan port tujuan (:80). Keterangan [**] [1:1000001:3] [*SINGLE SYN*] *Detected SYN packet* menandakan rule *Snort* mendeteksi paket SYN tunggal, yang sering dipakai dalam skenario serangan *SYN Flood*. Sumber serangan tampak berasal dari IP 192.168.89.3, yang kemungkinan adalah *attacker*. Alamat tujuan adalah *server*. Hasil ini menunjukkan bahwa *Snort* berhasil *real-time monitoring* trafik jaringan dan langsung mencatat semua percobaan koneksi *SYN*.



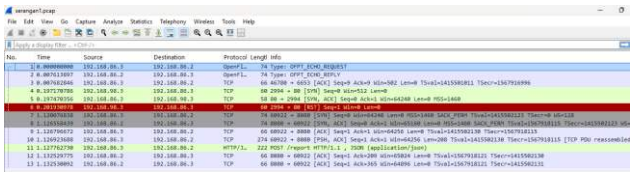
GAMBAR 13
(EKSEKUSI SCRIPT SNORT_DROP_STATS.PY UNTUK PROTEKSI DDOS SYN FLOOD)

Gambar 13 menunjukkan proses eksekusi *script Python snort_drop_stats.py* yang berfungsi sebagai penghubung otomatis antara hasil deteksi *Snort* dengan tindakan eksekusi aktif menggunakan *iptables* dan *ipset.* Pada tampilan terminal, *script* pertama kali menampilkan pesan "*Watcher snort_drop_stats started... monitoring Snort alerts*." sebagai tanda bahwa proses pemantauan log *alert Snort* sudah berjalan secara *real-time*. Setelah mendeteksi alamat *IP* penyerang, *script* secara otomatis menambahkan *IP* tersebut ke *ipset blacklist* dan membuat aturan *DROP* di *iptables*. Hal ini terlihat dari keluaran yang menyatakan "IP 192.168.98.2 ditambahkan ke *ipset blacklist*" dan "*Iptables rule DROP* ditambahkan untuk 192.168.98.2". Selain memblokir lalu lintas penyerang, *script* juga mengirimkan laporan ke *controller Ryu*, yang dikonfirmasi melalui pesan "Laporan terkirim ke Ryu... status: 200" sebagai tanda bahwa *REST API* menerima data dengan sukses.



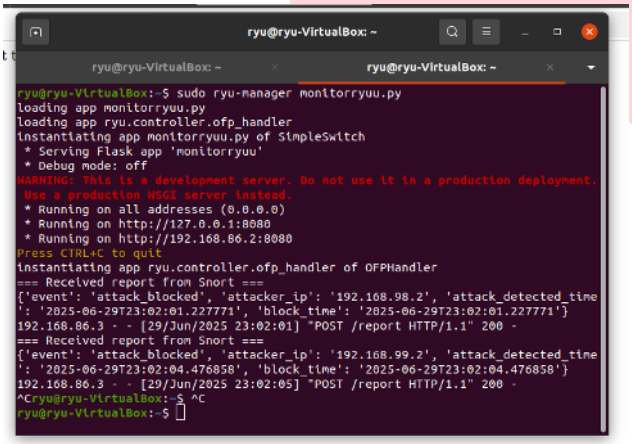
GAMBAR 14
(CAPTURE LOG IP YANG TELAH DI DROP OLEH SCRIPT)

Gambar 14 memperlihatkan hasil eksekusi yang terletak pada */var/log/snort_drop.log* pada terminal *server* yang menjalankan sistem *IDPS* berbasis *Snort*. Log ini merupakan hasil dari *flow* otomatis di mana *Snort* mendeteksi pola serangan *SYN Flood* melalui *rule* yang sudah dikonfigurasi, kemudian *script Python* membaca file *alert Snort* dan langsung memitigasi dengan memblokir *IP* penyerang. Selanjutnya, *script* mencatat aksi blokir ini ke file */var/log/snort_drop.log* sebagai bukti tindak responsif sistem *IDPS*. Dalam gambar 4.11, terlihat beberapa entri log yang menampilkan *timestamp* waktu pemblokiran, disertai informasi alamat IP sumber serangan. Isi nya pada tanggal 29 Juni 2025 pukul 19:59:52, sistem secara otomatis menambahkan IP. 192.168.98.3 ke daftar blokir (*blacklist*) melalui integrasi *ipset* dan *iptables*. Hal serupa terjadi pada IP lain, seperti 192.168.98.2 dan 192.168.99.2, yang juga tercatat telah diblokir pada waktu berbeda.



GAMBAR 15
(CAPTURE WIRESHARK SAAT SERANGAN SYN FLOOD MINIMAL DENGAN IDPS)

Gambar 15 menunjukkan bahwa terdapat interaksi komunikasi antara *host* dengan alamat *IP* 192.168.86.3 dan beberapa *host* lain, salah satunya adalah 192.168.98.3. Fokus utama ada pada *traffic* yang menggunakan protokol *TCP*, khususnya *port* tujuan 80, yang biasa digunakan untuk komunikasi *HTTP*. Baris yang ditandai merah yaitu pada baris ke-6 memperlihatkan adanya *TCP RST* (*Reset*), yaitu sinyal bahwa koneksi dihentikan secara paksa. Ini bisa menjadi indikasi bahwa terjadi upaya pemutusan koneksi akibat *traffic* yang tidak diharapkan bisa karena sistem mendeteksi aktivitas mencurigakan seperti serangan *SYN Flood*.



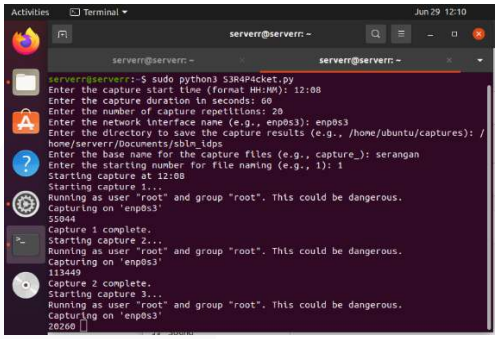
GAMBAR 16 (TERMINAL RYU MENERIMA FLOW DARI SNORT)

Gambar 16 menunjukkan tampilan terminal pada *controller Ryu* setelah menerima laporan serangan dari sistem *IDPS Snort* yang berjalan di *server*. Pada pengujian ini, *script Python* penghubung secara *otomatis* mengirimkan *HTTP POST request ke endpoint REST API Ryu* setiap kali *Snort* mendeteksi dan memblokir serangan *SYN Flood*. Dari *output* terminal pada gambar 25, terlihat empat entri laporan yang masing-masing berisi informasi detail berupa *event attack_blocked*, *IP* penyerang, waktu deteksi serangan (*attack_detected_time*), serta waktu tindakan pemblokiran (*block_time*). Setelah menerima data, *Ryu* mencetak respon status *HTTP* 200 yang menandakan bahwa laporan berhasil diterima dan diproses.

3. Serangan SYN Flood Serangan Flooding Tanpa IDPS

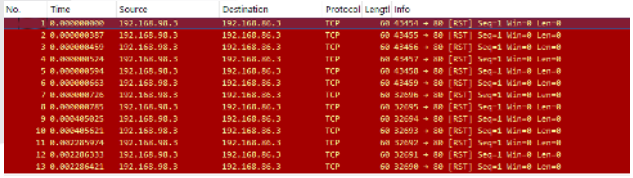
GAMBAR 17
(PENGUJIAN SERANGAN SYN FLOOD TANPA IDPS SERANGAN FLOODING)

Pada gambar 17 menunjukan pengujian serangan *SYN Flood Flooding* yang dilakukan kepada *PC Server* dengan alamat *IP* 192.168.86.3 sebagai target serangan tanpa menggunakan *IDPS* sebagai sistem perlindungan. Serangan dijalankan dari dua *attacker* untuk mensimulasikan serangan sehingga dapat membanjiri *server* dengan permintaan *SYN*. Dalam kondisi tanpa perlindungan *IDPS* atau mekanisme filter berbasis *firewall*, lalu lintas semacam ini dapat melewati jaringan tanpa terdeteksi dan berkontribusi pada beban *CPU server* secara drastis.



GAMBAR 18
(CAPTURE LALU LINTAS JARINGAN SAAT SYN FLOOD MODE FLOODING)

Pada gambar 18 menunjukkan pada PC *Server script python S3R4P4cket.py* sedang melakukan rekam lalu lintas jaringan ketika pengujian serangan dilakukan. *Script* ini mencatat semua paket yang melintasi *interface enp0s3* dan paket tersebut akan disimpan dalam format *.pcap* agar dapat di analisis.

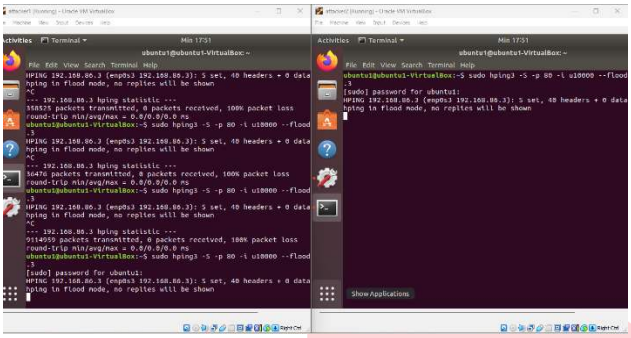


GAMBAR 19
(CAPTURE WIRESHARK SAAT SERANGAN SYN FLOOD MODE FLOODING TANPA IDPS)

Gambar 19 menunjukkan hasil tangkapan lalu lintas jaringan menggunakan *Wireshark* saat dilakukan simulasi serangan *SYN Flood* dari *IP* penyerang 192.168.98.3 menuju *server* target dengan alamat IP 192.168.86.3 pada *port* 80. Terlihat bahwa dalam rentang waktu yang sangat singkat, yaitu hanya sekitar 0,02 detik sejak paket pertama, penyerang sudah mengirimkan lebih dari sepuluh paket *TCP* secara beruntun. Semua paket tersebut memiliki *TCP flag* yang ditandai sebagai [*RST*], yang berarti *Reset* koneksi. Karena pengujian ini *IDPS* belum di aktifkan, maka tidak ada

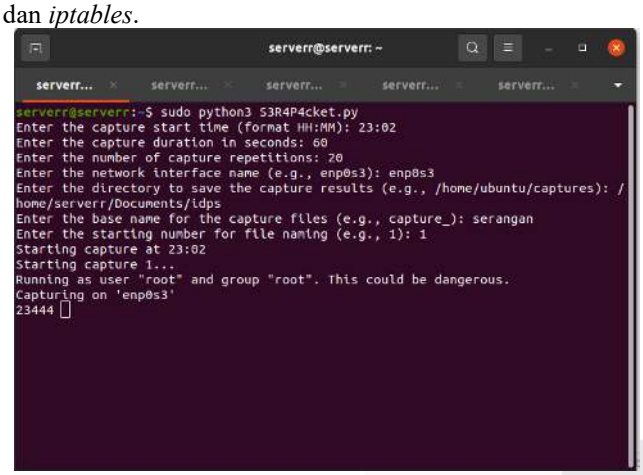
mekanisme untuk mendeteksi dan memfilter atau memblokir paket *SYN* yang masuk. Kondisi ini berpotensi menyebabkan penurunan performa layanan.

4. Serangan SYN Flood Serangan Flooding Dengan IDPS



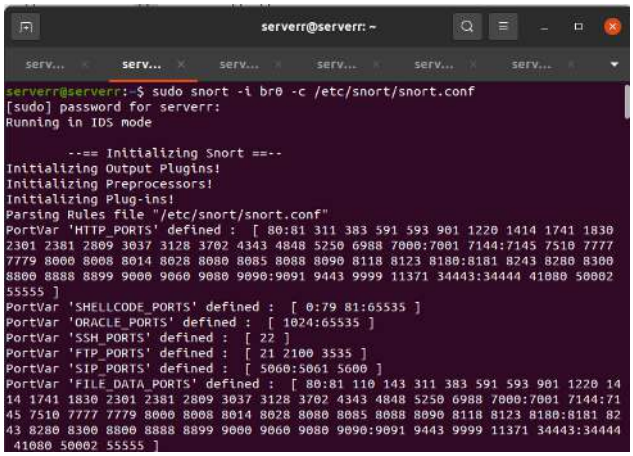
GAMBAR 20
(PENGUJIAN SERANGAN SYN FLOOD FLOODING DENGAN IDPS)

Pada gambar 20 memperlihatkan pengujian serangan *SYN Flood* yang dilakukan kepada *PC Server* dengan alamat IP 192.168.86.3 sebagai target serangan dengan sistem perlindungan *IDPS* yang telah di aktifkan. Sistem pada skenario ini telah dilengkapi perlindungan *IDPS* aktif berbasis *Snort*. Sistem *IDPS* mendeteksi pola serangan *SYN Flood*, kemudian secara otomatis memfilter, menganalisis, dan memblokir *IP* sumber serangan dengan integrasi *ipset* dan *iptables*.



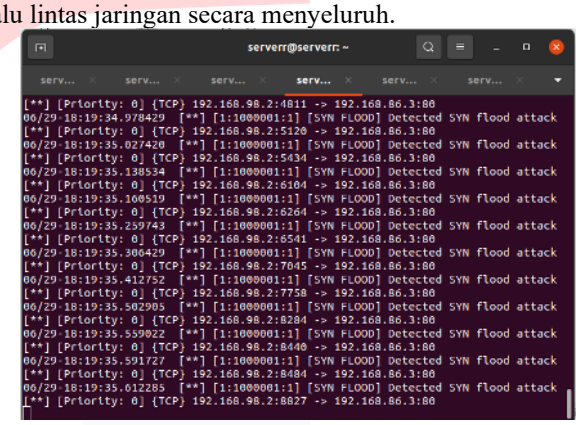
GAMBAR 21
(REKAM LALU LINTAS SAAT SERANGAN SYN FLOOD FLOODING DENGAN IDPS AKTIF)

Pada gambar 21 menunjukkan pada *PC Server script python S3R4P4cket.py* sedang melakukan rekam lalu lintas jaringan ketika pengujian serangan dilakukan. *Script* ini mencatat semua paket yang melintasi *interface enp0s3* dan paket tersebut akan disimpan dalam format *.pcap* agar dapat di analisis.



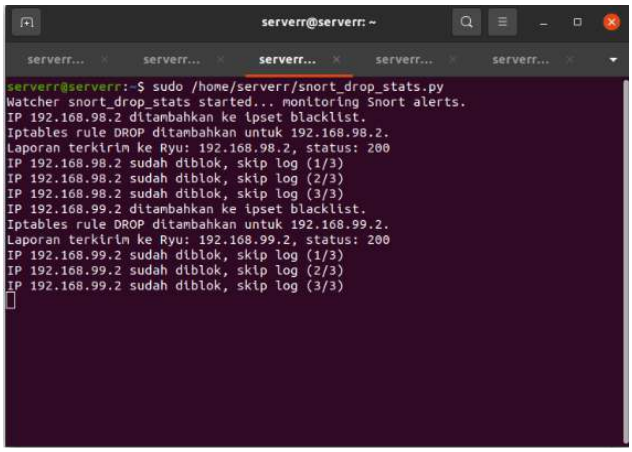
GAMBAR 22
(TERMINAL SAAT SNORT DI AKTIFKAN)

Gambar 22 menunjukkan proses inisialisasi *Snort* yang dijalankan pada mode *IDS* (*Intrusion Detection System*) di *server*. Proses inisialisasi ini merupakan tahap awal yang penting karena memastikan seluruh komponen pendukung *Snort* sudah dimuat dan siap untuk melakukan pemantauan lalu lintas jaringan secara menyeluruh.



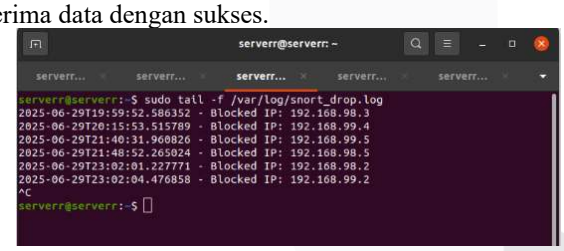
GAMBAR 23
(LOG ALERT SAAT SERANGAN SYN FLOOD MODE FLOODING DENGAN IDPS)

Gambar 23 menunjukkan tampilan keluaran log hasil monitoring *file /var/log/snort/alert* yang merekam aktivitas deteksi serangan *SYN Flood.* Snort secara otomatis mencatat setiap paket yang terindikasi sebagai bagian dari pola serangan SYN Flood, ditandai dengan pesan alert bertuliskan [SYN FLOOD] Detected SYN flood attack. Berdasarkan data yang ditampilkan pada gambar 4.20, dapat dilihat bahwa serangan berasal dari alamat IP 192.168.98.2 dengan intensitas pengiriman paket yang sangat tinggi dalam selang waktu yang rapat. Hal ini menunjukkan kemampuan Snort dalam mendeteksi serangan secara real-time dan mencatat setiap upaya koneksi yang mencurigakan ke dalam log alert.



GAMBAR 24
(EKSEKUSI SCRIPT SNORT_DROP_STATS.PY UNTUK PROTEKSI DDOS)

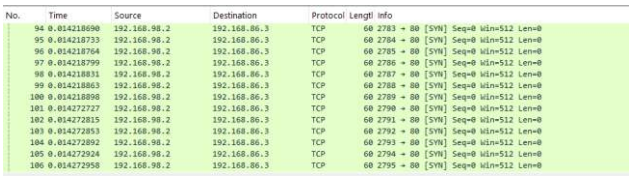
Gambar 24 menunjukkan proses eksekusi *script Python snort_drop_stats.py* yang berfungsi sebagai penghubung otomatis antara hasil deteksi *Snort* dengan tindakan eksekusi aktif menggunakan *iptables* dan *ipset*. Pada tampilan terminal, *script* pertama kali menampilkan pesan "*Watcher snort_drop_stats started... monitoring Snort alerts*." sebagai tanda bahwa proses pemantauan log *alert Snort* sudah berjalan secara *real-time*. Setelah mendeteksi alamat *IP* penyerang, *script* secara otomatis menambahkan *IP* tersebut ke *ipset blacklist* dan membuat aturan *DROP* di iptables. Hal ini terlihat dari keluaran yang menyatakan "IP 192.168.98.2 ditambahkan ke *ipset blacklist*" dan "*Iptables rule DROP* ditambahkan untuk 192.168.98.2". selain memblokir lalu lintas penyerang, *script* juga mengirimkan laporan ke *controller Ryu*, yang dikonfirmasi melalui pesan "Laporan terkirim ke *Ryu*... status: 200" sebagai tanda bahwa *REST API* menerima data dengan sukses.



GAMBAR 25
(CAPTURE IP YANG TELAH DI DROP SCRIPT)

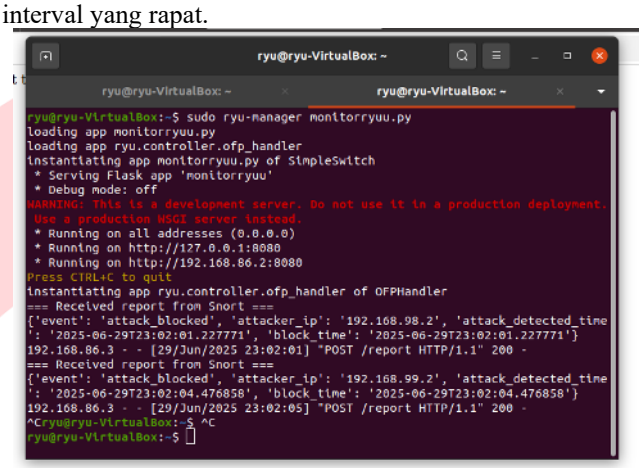
Gambar 25 memperlihatkan hasil eksekusi yang terletak pada */var/log/snort_drop.log* pada terminal *server* yang menjalankan sistem *IDPS* berbasis *Snort*. Log ini merupakan hasil dari *flow* otomatis di mana *Snort* mendeteksi pola serangan *SYN Flood* melalui *rule* yang sudah dikonfigurasi, kemudian *script Python* membaca *file alert Snort* dan langsung memitigasi dengan memblokir *IP* penyerang. Selanjutnya, *script* mencatat aksi blokir ini ke *file /var/log/snort_drop.log* sebagai bukti tindak responsif sistem *IDPS*.

Dalam gambar 4.22, terlihat beberapa entri log yang menampilkan *timestamp* waktu pemblokiran, disertai informasi alamat IP sumber serangan. Data pada tanggal 29 Juni 2025 pukul 19:59:52, sistem secara otomatis menambahkan *IP* 192.168.98.3, 192.168.98.2, dan 192.168.99.2 ke daftar blokir (*blacklist*) melalui integrasi *ipset* dan *iptables*.



GAMBAR 26
(CAPTURE WIRESHARK SAAT SYNFLOOD FLOODING DENGAN IDPS AKTIF)

Gambar 26 menunjukkan hasil *capture wireshark* saat serangan *SYN Flood flooding* dilakukan ke server 192.168.86.3 oleh *attacker* 192.168.98.2, setelah *IDPS* yaitu snort telah aktif. Terlihat bahwa *attacker* terus menerus mengirimkan paket *TCP* dengan *flag [SYN]* dalam waktu interval yang rapat.

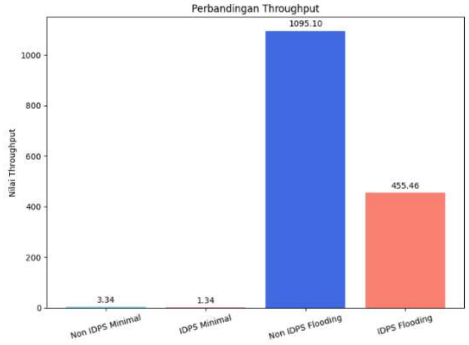


GAMBAR 27
(TERMINAL RYU MENERIMA FLOW DARI SNORT)

Gambar 27 menunjukkan tampilan terminal pada *controller Ryu* setelah menerima laporan serangan dari sistem *IDPS Snort* yang berjalan di *server*. Pada pengujian ini, *script Python* penghubung secara *otomatis* mengirimkan *HTTP POST request ke endpoint REST API Ryu* setiap kali *Snort* mendeteksi dan memblokir serangan *SYN Flood*. Dari *output* terminal pada gambar 25, terlihat empat entri laporan yang masing-masing berisi informasi detail berupa *event attack_blocked*, *IP* penyerang, waktu deteksi serangan (*attack_detected_time*), serta waktu tindakan pemblokiran (*block_time*). Setelah menerima data, *Ryu* mencetak respon status *HTTP* 200 yang menandakan bahwa laporan berhasil diterima dan diproses.

B. Pengukuran Quality of Service (QoS)

1. Throughput

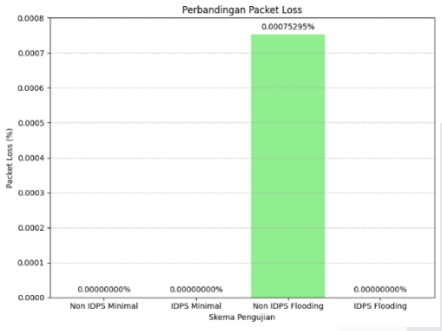


GAMBAR 28
(PERBANDINGAN THROUGHPUT PADA SETIAP PENGUJIAN)

Grafik menunjukkan perbandingan *throughput*. Dari keempat skenario tersebut, nilai *throughput* tertinggi berada pada kondisi *Non IDPS Flooding* dengan nilai sebesar 1095,10 bit/s. Pada skenario *IDPS Flooding*, *throughput* turun secara signifikan menjadi 455,45 bit/s. Penurunan ini menjadi indikator bahwa *IDPS* bekerja dengan baik dalam menyaring dan memblokir sebagian besar paket berbahaya yang dikirimkan saat serangan *SYN Flood* berlangsung. Hal ini memperlihatkan bahwa penerapan *IDPS* pada kondisi *flooding* memberikan perlindungan yang cukup baik.

Sementara itu, pada kondisi serangan minimal, *throughput* yang dihasilkan jauh lebih kecil, yaitu 3,33 bit/s untuk *Non IDPS* Minimal dan turun menjadi 1,34 bit/s saat *IDPS* diaktifkan. Secara keseluruhan, grafik ini memperlihatkan bahwa penerapan *IDPS* memiliki dampak signifikan dalam mengontrol lalu lintas jaringan yang berbahaya. Perbandingan antara kondisi dengan dan tanpa *IDPS* pada dua skenario serangan (minimal dan *flooding*) memperlihatkan pola yang konsisten.

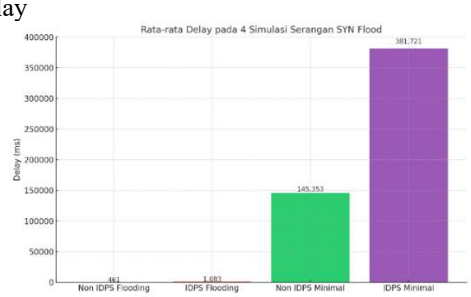
2. Packet Loss



GAMBAR 29
(PERBANDINGAN PACKET LOSS PADA SETIAP PENGUJIAN)

Berdasarkan hasil pengukuran *packet loss* pada masing-masing skenario, terlihat bahwa *Non IDPS* Minimal, *IDPS* Minimal, dan *IDPS Flooding* semuanya menunjukkan nilai *packet loss* sebesar 0,000000%. Artinya, dalam ketiga skenario tersebut tidak ada paket yang hilang saat proses transmisi data berlangsung. Namun, pada skenario *Non IDPS Flooding*, ditemukan *packet loss* sebesar 0,00075295%. Penyebabnya adalah tidak adanya sistem deteksi dan pencegahan (*IDPS*) yang aktif saat serangan *DDoS SYN Flood* diluncurkan. Karena tidak ada mekanisme perlindungan, jaringan mengalami beban yang tinggi, sehingga sebagian kecil paket tidak dapat diproses dan akhirnya hilang. Secara keseluruhan, dapat disimpulkan bahwa penggunaan *IDPS* sangat efektif dalam meminimalkan risiko kehilangan data selama transmisi jaringan.

3. Delay



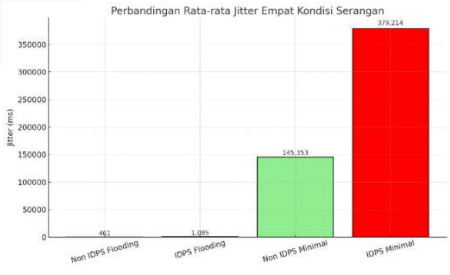
GAMBAR 30
(PERBANDINGAN NILAI DELAY PADA SETIAP PENGUJIAN)

Berdasarkan hasil pengukuran rata-rata *delay* pada keempat skenario, terlihat perbedaan yang cukup signifikan antara kondisi serangan minimal dan kondisi serangan *flooding*, baik dengan maupun tanpa sistem *IDPS*. Skenario *Non IDPS* Minimal mencatat rata-rata *delay* sebesar 145.352 ms, sedangkan *IDPS* Minimal mengalami kenaikan *delay* yang sangat tinggi hingga mencapai 381.720 ms. Hal ini menunjukkan bahwa meskipun trafik serangan yang dilakukan tidak dalam mode *flooding*. Proses filtering dan pemrosesan paket oleh *Snort* serta *rule iptables/ipset* yang berjalan secara *real-time* membuat waktu propagasi paket menjadi lebih lambat, sehingga *delay* meningkat secara drastis dibandingkan kondisi tanpa proteksi.

Non IDPS Flooding memiliki rata-rata delay sekitar 460 ms, sementara *IDPS Flooding* menunjukkan nilai 1.082 ms.. Jika dibandingkan antara penggunaan *IDPS* dan tanpa *IDPS* pada skenario serangan flooding, terlihat bahwa penggunaan *IDPS* tetap memberikan efek peningkatan delay hampir dua kali lipat. *Delay* yang tercatat pada *IDPS* Flooding rata-rata mencapai sekitar 1 detik, sedangkan *Non IDPS Flooding* tetap berada di bawah 500 ms.

Secara keseluruhan, hasil pengujian ini menunjukkan bahwa penggunaan *IDPS* berbasis *Snort* yang dikombinasikan *iptables* dan *ipset* memang efektif dalam mengeksekusi serangan, tetapi konsekuensinya adalah peningkatan *delay* yang cukup signifikan, terutama ketika serangan dilakukan secara terus-menerus meskipun volumenya kecil.

4. Jitter



GAMBAR 31
(PERBANDINGAN NILAI JITTER PADA SETIAP PENGUJIAN)

Diagram batang ini menunjukkan perbandingan rata-rata *jitter* dari empat skenario pengujian serangan *SYN Flood* terhadap jaringan. Pada skenario *Non IDPS Flooding*, nilai *jitter* rata-rata tercatat 460,72 ms. Berbeda halnya ketika *IDPS* diaktifkan pada skenario *flooding*, nilai *jitter* naik menjadi 1084,73 ms. Kenaikan ini terjadi karena *IDPS* harus

memproses setiap paket dalam *volume* trafik yang besar, melakukan inspeksi, dan memutuskan apakah paket itu ancaman atau bukan.

Skenario *Non IDPS Minimal* di mana *jitter* melonjak sangat tinggi hingga 145.352 ms. Terakhir, skenario *IDPS* Minimal menunjukkan rata-rata jitter tertinggi yaitu 379.214 ms. Ini semakin memperjelas perbedaan efek serangan ketika *IDPS* aktif

V. KESIMPULAN

Penelitian ini berhasil merancang dan mengimplementasikan sistem keamanan jaringan berbasis *Software-Defined Network (SDN*) dengan mengintegrasikan Snort sebagai *Intrusion Detection and Prevention System* (IDPS) dan *Ryu Controller* untuk mendeteksi serta memblokir serangan *DDoS SYN Flood* secara otomatis. Sistem ini bekerja dengan memanfaatkan rule Snort untuk mendeteksi pola serangan secara real-time. Pengujian dilakukan selama 20 menit pada dua skenario serangan *SYN Flood* serangan *flooding* dan serangan minimal dengan kondisi menggunakan IDPS dan tanpa IDPS. Berdasarkan pengukuran *Quality of Service* (QoS), pada serangan flooding, throughput turun signifikan dari ±1095 bit/s menjadi ±455 bit/s saat IDPS diaktifkan. Parameter delay dan jitter cenderung meningkat dengan IDPS, terutama pada serangan minimal, di mana delay rata-rata tercatat di atas 380.000 ms akibat proses inspeksi paket. Meski demikian, nilai *packet loss* tetap sangat rendah, tidak melebihi 1%, sehingga konektivitas jaringan tetap terjaga walaupun dalam kondisi serangan. Secara keseluruhan, integrasi Snort dengan SDN terbukti efektif dalam mendeteksi dan memitigasi serangan *SYN Flood* secara real-time dengan keberhasilan penyaringan trafik berbahaya yang signifikan, meskipun berdampak pada peningkatan delay dan jitter yang perlu dioptimalkan agar tidak mengganggu kualitas layanan bagi pengguna normal.

REFERENSI

[1] Cindy Mutia Annur, "Individu Pengguna Internet Global Tembus 5,35 Miliar pada Januari 2024," databoks. Accessed: Jun. 15, 2024. [Online]. Available: https://databoks.katadata.co.id/datapublish/2024/02/08/individu-pengguna-internet-global-tembus-535-miliar-pada-januari-2024

[2] "APJII Jumlah Pengguna Internet Indonesia Tembus 221 Juta Orang." Accessed: Jun. 15, 2024. [Online]. Available: https://apjii.or.id/berita/d/apjii-jumlah-pengguna-internet-indonesia-tembus-221-juta-orang

[3] Muqowam Akhmad, "Indonesia Peringkat ke-2 Dunia Kasus Kejahatan Siber," Republika. Accessed: Jul. 23, 2025. [Online]. Available: https://news.republika.co.id/berita/nmjajy/indonesia-peringkat-ke2-dunia-kasus-kejahatan-siber

[4] "Jenis-Jenis Serangan Siber di Era Digital." Accessed: Jun. 15, 2024. [Online]. Available: https://bpptik.kominfo.go.id/Publikasi/detail/jenis-jenis-serangan-siber-di-era-digital

[5] "3 Juta Serangan Siber Ancam Pengguna Internet di RI Sepanjang Q1 2025," CNN Indonesia. Accessed: Jul. 25, 2025. [Online]. Available: https://www.cnnindonesia.com/teknologi/20250502154407-192-1225071/3-juta-serangan-siber-ancam-pengguna-internet-di-ri-sepanjang-q1-2025

[6] T. Vimy *et al.*, "Ancaman Serangan Siber Pada Keamanan Nasional Indonesia," *Jurnal Kewarganegaraan*, vol. 6, no. 1, 2022.

[7] S. Prathibha *et al.*, "Detection Methods for Software Defined Networking Intrusions (SDN)," in *Proceedings - IEEE International Conference on Advances in Computing, Communication and Applied Informatics, ACCAI 2022*, Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ACCAI53970.2022.9752574.

[8] S. Hanadwiputra, Subadri, and D. Prawinarko, "Implementasi Konsep Software Defined Networking (Sdn) Wide Area Network (Wan) Pada Mikrotik Dengan Python 3," *JUPITER Jurnal Teknologi Informatika & Komputer*, vol. 4, no. 2, 2023.

[9] H. Alamsyah, Riska, and A. Al Akbar, "Analisa Keamanan Jaringan Menggunakan Network Intrusion Detection and Prevention System," *JOINTECS (Journal of Information Technology and Computer Science)*, vol. 5, no. 1, p. 17, 2020, doi: 10.31328/jointecs.v5i1.1240.

[10] A. Yasin and I. Mohidin, "Dampak Serangan DDoS pada Software Based Openfow Switch di Perangkat HG553," *Jurnal Technopreneur (JTech)*, vol. 6, no. 2, p. 72, Nov. 2018, doi: 10.30869/jtech.v6i2.206.

[11] M. , S. A. L. A. Kalee, "Klasifikasi Serangan Distributed Denial-of-Service (DDoS) menggunakan Metode Data Mining Naïve Bayes," 2022. [Online]. Available: http://j-ptiik.ub.ac.id

[12] D. Scholz, S. Gallenmüller, H. Stubbe, B. Jaber, M. Rouhi, and G. Carle, "Me Love (SYN-)Cookies: SYN Flood Mitigation in Programmable Data Planes," 2020.

[13] Michell Kaleel, "Comparative Analysis Of 4.5g Lte Internet Network Quality In Manado City," *Jurnal Teknik Informatika*, 2022.