ANALISIS PENGGUNAAN *RATE LIMITING* UNTUK MENGURANGI DAMPAK SERANGAN ICMP *FLOOD* PADA *SOFTWARE-DEFINED NETWORK*

1st Dr. Mochamad Teguh Kurniawan, S.T., M.T.
Universitas Telkom
S1 Sistem Informasi
Bandung, Indonesia
teguhkurniawan@telkomuniversity.ac.id

2nd Muhammad Fathinuddin, S.SI., M.T
Universitas Telkom
S1 Sistem Informasi
Bandung, Indonesia
muhammadfathinuddin@telkomuniversity.ac.id

3rd Rafif Rasalhague
Universitas Telkom
S1 Sistem Informasi
Bandung, Indonesia
rasalhague@student.telkomuniversity.ac.id

***Abstrak* — Perkembangan *Software-Defined Network* (SDN) memberikan fleksibilitas tinggi dalam pengelolaan lalu lintas data melalui kontrol terpusat, namun juga memunculkan tantangan keamanan baru, seperti *serangan Internet Control Message Protocol* (ICMP) *Flood*. Penelitian ini menganalisis penerapan algoritma *Support Vector Machine* (SVM) untuk mendeteksi lalu lintas anomali, serta teknik *rate limiting* sebagai mitigasi terhadap serangan ICMP *Flood* pada SDN berbasis POX *Controller*. Simulasi dilakukan di Mininet dengan lima skenario (dua penyerang tanpa mitigasi, tiga hingga lima penyerang dengan mitigasi), mengirim hingga 100.000 paket ICMP berukuran 5.000 byte dengan interval 1–2,5 ms. Parameter yang diuji mencakup *packet length*, ICMP *count*, *packet loss*, dan *round-trip time* (RTT). Tanpa mitigasi, durasi gangguan mencapai 1.323 detik dan *false positive* (FP) sebanyak 23.134, menunjukkan kegagalan dalam membedakan *traffic* sah dari *flood*. Dengan mitigasi, lonjakan *delay* berkurang menjadi 45–94 detik dan FP turun ke kisaran 1.396–2.589. Selain itu, *packet loss* tetap di bawah 72%, dan akurasi deteksi SVM mencapai 93,48%–95,82%. Hasil ini menunjukkan bahwa kombinasi SVM dan *rate limiting* efektif menekan dampak serangan tanpa mengganggu lalu lintas normal, menjaga kestabilan jalur komunikasi meskipun berada di bawah tekanan *flood*.**

***Kata kunci*— *Software-Defined Network*, ICMP *Flood*, DDoS, POX *Controller*, *Rate Limiting*, Keamanan Jaringan**

I. PENDAHULUAN

Seiring pesatnya perkembangan teknologi jaringan, kebutuhan akan sistem komunikasi data yang andal dan aman semakin mendesak. *Software-Defined Network* (SDN) muncul sebagai paradigma baru yang memisahkan fungsi kontrol jaringan dari perangkat keras (*hardware*), sehingga memungkinkan manajemen lalu lintas dilakukan secara terpusat melalui *controller* [1]. Namun, keunggulan ini juga menjadi celah kelemahan baru, di mana titik kontrol terpusat rawan menjadi sasaran serangan siber seperti *Distributed Denial of Service* (DDoS), salah satunya adalah ICMP *Flood*[2].

Serangan ICMP *Flood* bekerja dengan membanjiri jaringan menggunakan paket ICMP dalam jumlah besar, membebani *controller* hingga menurunkan performa bahkan dapat melumpuhkan jaringan [3]. Salah satu solusi untuk menghadapi serangan semacam ini adalah dengan penerapan metode rate limiting yang berfungsi membatasi laju paket [4]. Di sisi lain, deteksi dini berbasis machine learning seperti *Support Vector Machine* (SVM) juga diperlukan agar sistem mampu mengidentifikasi pola lalu lintas anomali secara adaptif [5], [6].

Penelitian ini bertujuan menganalisis penerapan SVM sebagai deteksi awal serangan ICMP *Flood* serta implementasi *rate limiting* pada SDN berbasis POX *Controller*. Simulasi dilakukan menggunakan Mininet dengan skenario lalu lintas normal dan serangan, serta evaluasi parameter performa meliputi *packet length*, ICMP *count*, *packet loss*, dan *round-trip time* (RTT). Dengan pendekatan ini, diharapkan tercipta sistem keamanan SDN yang lebih adaptif dan efisien.

II. KAJIAN TEORI

II.1 Jaringan Komputer

Jaringan komputer adalah sekumpulan perangkat komputer yang saling terhubung melalui media komunikasi untuk bertukar data dan sumber daya, seperti printer, penyimpanan, dan aplikasi [7]. Jaringan komputer memfasilitasi pertukaran data baik secara lokal maupun global melalui Internet. Seiring perkembangan teknologi, jaringan komputer juga mendukung beragam layanan modern seperti *cloud computing*, *Internet of Things* (*IoT*), dan sistem *big data*. Struktur jaringan komputer biasanya diatur dalam topologi tertentu, seperti topologi *tree*, *star*, atau *mesh*, tergantung kebutuhan skalabilitas dan kecepatan transfer data [8].

II.2 *Software-Defined Network* (SDN)

Software-Defined Network (*SDN*) adalah paradigma arsitektur jaringan yang memisahkan *control plane* (logika kontrol) dari *data plane* (pengiriman data) [1]. Pendekatan ini memungkinkan pengelolaan lalu lintas jaringan secara terpusat melalui sebuah *controller* yang dapat diprogram secara dinamis. Keunggulan SDN terletak pada fleksibilitas manajemen, skalabilitas, dan kemudahan pengaturan kebijakan keamanan. Dengan kontrol terpusat, SDN mendukung pengembangan kebijakan otomatis, seperti

firewall rules, *load balancing*, hingga penanganan serangan siber. Namun, arsitektur terpusat juga membuka celah kelemahan: apabila *controller* diserang, seluruh jaringan dapat lumpuh [9].

II.3 *Distributed Denial of Service* (DDoS)

Serangan *Distributed Denial of Service* (*DDoS*) adalah upaya melumpuhkan layanan jaringan dengan membanjiri target menggunakan lalu lintas data dari banyak sumber secara bersamaan [10]. Dalam konteks SDN, serangan DDoS menjadi lebih berbahaya karena dapat langsung menargetkan *controller* sebagai titik pusat kontrol. Hal ini dapat menyebabkan bottleneck pada *controller*, menurunkan performa, dan menghentikan proses komunikasi normal di jaringan. Oleh karena itu, deteksi dini dan mitigasi serangan DDoS sangat krusial untuk menjaga ketersediaan layanan [1].

II.4 *Internet Control Message Protocol* (ICMP) *Flood*

ICMP *Flood* adalah salah satu jenis serangan volumetrik DDoS yang memanfaatkan protokol ICMP untuk membanjiri target dengan pesan ping berulang [2]. Dalam skenario normal, ping digunakan untuk memeriksa konektivitas antar host. Namun, pada serangan ICMP *Flood*, penyerang mengirim paket ICMP dalam volume besar dengan interval cepat, memaksa target memproses setiap paket hingga kehabisan sumber daya. Dalam arsitektur SDN, serangan ICMP *Flood* dapat membebani *controller* dengan lalu lintas berlebih, mengakibatkan penurunan *throughput*, *packet loss*, dan delay yang signifikan [4].

II.5 POX *Controller*

POX *Controller* adalah salah satu *controller open-source* yang populer digunakan dalam riset SDN karena fleksibilitasnya dalam mendukung protokol OpenFlow dan kemudahan penyesuaian [11]. POX ditulis menggunakan Python, sehingga memudahkan pengembangan modul deteksi serangan, pencatatan log, dan penerapan kebijakan *flow entries* secara *real-time*. Dengan POX, administrator dapat memprogram *flow rules* seperti *blocking* IP, *rate limiting*, hingga monitoring lalu lintas tanpa harus mengonfigurasi perangkat keras secara manual.

II.6 *Support Vector Machine* (SVM)

Support Vector Machine (SVM) adalah algoritma *machine learning* untuk tugas klasifikasi biner, seperti membedakan antara lalu lintas normal dan anomali [5], [6]. SVM bekerja dengan membangun sebuah *hyperplane* untuk memisahkan data dari dua kelas dengan margin maksimal. Dalam konteks keamanan jaringan, SVM digunakan untuk mendeteksi pola lalu lintas mencurigakan, seperti lonjakan paket ICMP Flood. SVM dikenal memiliki akurasi tinggi dan mampu beradaptasi pada dataset dengan pola distribusi yang jelas.

II.7 *Rate Limiting*

Rate limiting adalah teknik membatasi jumlah paket data yang diperbolehkan melewati jaringan dalam satuan waktu tertentu [4]. Dalam mitigasi serangan ICMP Flood pada SDN, *rate limiting* diterapkan melalui *controller* untuk mengatur laju paket ICMP agar tidak membebani *controller*. Jika jumlah paket melebihi ambang batas (*threshold*), sistem akan memblokir IP sumber atau menjatuhkan paket ke jalur *blackhole*. Penerapan *rate limiting* membantu menstabilkan performa jaringan dan mencegah *bottleneck* akibat serangan volumetrik.

II.8 *Confusion Matrix*

Confusion matrix adalah metode evaluasi model klasifikasi dengan membandingkan label prediksi dan label aktual [6]. Matriks ini menampilkan jumlah *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Pada deteksi serangan ICMP *Flood*, *confusion matrix* membantu mengevaluasi akurasi SVM dalam memisahkan lalu lintas normal dan serangan. TP menunjukkan serangan yang berhasil terdeteksi, TN menunjukkan lalu lintas normal yang dikenali dengan benar, FP adalah lalu lintas normal yang keliru terdeteksi sebagai serangan, dan FN adalah serangan yang tidak terdeteksi.

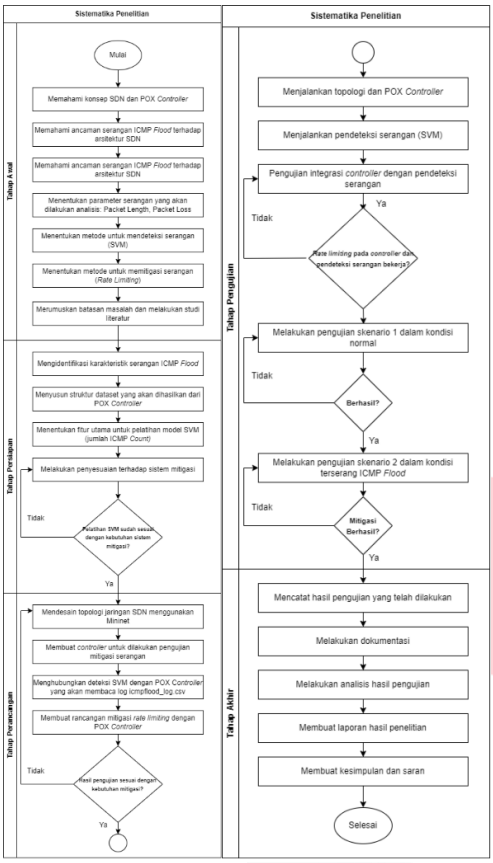
II.9 *Round Trip Time* (RTT)

Round Trip Time (*RTT*) adalah waktu yang dibutuhkan paket data untuk melakukan perjalanan pulang pergi dari pengirim ke penerima dan kembali lagi [3]. RTT menjadi indikator performa jaringan, terutama dalam mendeteksi *delay* akibat serangan DDoS. Serangan ICMP *Flood* dapat meningkatkan RTT secara drastis karena jalur komunikasi dibebani paket ping berlebih. Pengukuran RTT membantu mengevaluasi efektivitas sistem deteksi dan mitigasi. Dalam penelitian ini, parameter RTT diukur pada skenario lalu lintas normal dan lalu lintas dengan serangan ICMP *Flood*.

III. METODE

III.1 Sistematika Penyelesaian Masalah

Gambar III.1 merupakan *flowchart* untuk sistematika penyelesaian masalah yang digunakan.



Gambar III. 1 Sistematika Penyelesaian Masalah

Penelitian dimulai dengan memahami konsep dasar SDN dan cara kerja POX *Controller* sebagai pengatur lalu lintas data. Karakter serangan ICMP *Flood* dianalisis bersama dampaknya pada performa, fokus pada parameter *ICMP count* dan *packet loss*. Dua pendekatan *diterapkan*: deteksi serangan dengan SVM dan mitigasi dengan *rate limiting*.
Pada tahap persiapan, lalu lintas hasil serangan diidentifikasi untuk membentuk dataset yang direkam otomatis ke log oleh POX *Controller*. Fitur untuk pelatihan SVM ditetapkan, lalu logika mitigasi disusun agar respons sistem adaptif. Model divalidasi, dan disesuaikan ulang jika belum sesuai.
Rancangan sistem mencakup topologi *tree* di Mininet, *controller* POX untuk *routing* dan deteksi, serta modul *rate limiting* yang memblokir IP penyerang otomatis. Semua komponen diintegrasi dan diuji ulang agar berfungsi.
Pengujian dilakukan pada dua skenario: lalu lintas normal dan lalu lintas diserang. Sistem dioperasikan bersama deteksi SVM, lalu dicek fungsi *rate limiting* aktif saat serangan terdeteksi.

III.2 Metode Evaluasi

Metode evaluasi bertujuan mengukur akurasi deteksi dan efektivitas mitigasi pada SDN dengan POX *Controller*. Penilaian mencakup dua aspek: performa deteksi SVM dan kinerja *rate limiting* dalam membatasi serangan ICMP *Flood*.
Pada evaluasi deteksi, model diuji untuk memisahkan lalu lintas normal dan serangan. Hasilnya dibagi menjadi beberapa kategori: serangan yang berhasil terdeteksi (TP), lalu lintas normal yang salah dikira serangan (FP), lalu lintas normal yang dikenali dengan benar (TN), dan serangan yang tidak terdeteksi (FN). Akurasi dihitung dengan rumus:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

Hasil akan divisualisasikan dalam *confusion matrix* dan laporan klasifikasi.
Selanjutnya, evaluasi mitigasi dilakukan setelah serangan aktif selama 15 detik. Di sini, *controller* akan menjalankan tindakan otomatis. Efektivitasnya dilihat dari perubahan pada lalu lintas *flood*, kestabilan jaringan normal, dan ketepatan aktivasi mitigasi. Pengujian dilakukan dalam dua kondisi: saat lalu lintas normal untuk mengukur potensi kesalahan deteksi, dan saat serangan aktif menggunakan *hping3* untuk melihat apakah sistem bisa merespons dengan benar.
Semua data berupa log, grafik, dan laporan disimpan untuk dianalisis pada bagian evaluasi.

IV. ANALISIS DAN PERACANGAN

IV.1 Spesifikasi Perangkat

Pada tahap penelitian ini, dilakukan penentuan spesifikasi perangkat yang akan digunakan dalam melakukan pengujian simulasi pada SDN. Semua perangkat harus terpasang dan berjalan dengan baik untuk menjalankan simulasi.

IV.1.1 Spesifikasi Perangkat Lunak (*Software*)

Penelitian ini didukung perangkat lunak berupa OS virtual, emulator jaringan, SDN *controller*, bahasa pemrograman, serta pustaka *machine learning* dan visualisasi. Semua komponen ini diperlukan untuk merancang, menjalankan, dan mengevaluasi sistem deteksi dan mitigasi di jaringan SDN dengan POX *Controller*.

Perangkat Lunak	Versi	Fungsi
Ubuntu Linux	20.04.6 LTS (64 bit)	Sistem operasi pada mesin virtual untuk menjalankan seluruh simulasi.
Mininet	2.3.0	Emulator jaringan untuk membuat topologi SDN.
POX *Controller*	POX Git (Python 2.7 *Environment*)	Pengontrol SDN yang dikustomisasi untuk deteksi dan mitigasi.
Python	2.7.18 (POX), 3.8.10 (SVM)	Pengontrol SDN yang dikustomisasi untuk *controller* dan model SVM.
Scikit-learn	0.24	*Library machine learning* untuk pelatihan model SVM.
Matplotlib & Seaborn	3.x & 0.11	Visualisasi *confusion matrix* dan hasil klasifikasi.
Hping3	3.20051105	Melakukan simulasi serangan ICMP *Flood*.

Tabel IV. 1 Komponen *Software*

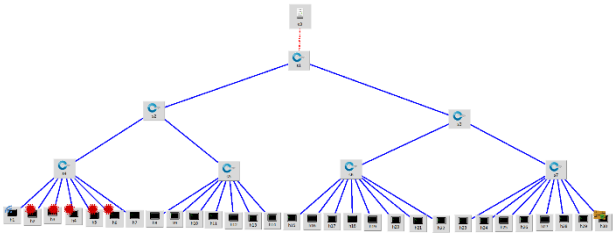
IV.2 Alur Perancangan

Bagian ini merangkum tahap perancangan deteksi dan mitigasi ICMP *Flood* pada SDN dengan POX *Controller* dan SVM. Proses dimulai dari pembuatan topologi *tree* (7 *switch*, 30 *host*), pengaturan peran *host*, hingga uji koneksi dan fungsi *controller*.
POX *Controller* disiapkan untuk mencatat trafik, mendeteksi serangan, dan memblokir lewat *rate limiting*. Model SVM

memproses log untuk klasifikasi, dievaluasi dengan *confusion matrix*. Semua tahap diatur berurutan agar sistem dapat mendeteksi dan merespons serangan secara terpadu.

IV.2.1 Topologi Jaringan dan Klasifikasi *Host*

Pembuatan topologi jaringan dilakukan pada emulator Mininet dengan bentuk topologi *tree* yang terdiri dari 1 *controller*, 7 *switch*, dan 30 *host*.



Gambar IV. 1 Topologi Jaringan

Topologi *tree* pada Gambar IV.1 tersebut memiliki komponen dengan fungsinya masing-masing yang tertera pada Tabel IV-3.

Komponen	Label	Fungsi
Controller	c0	Mengelola seluruh aliran lalu lintas jaringan SDN. Terhubung langsung ke *core switch* (s1).
Switch 1	s1	Switch pusat yang menghubungkan *controller* ke 2 jalur distribusi utama (s2 dan s3).
Switch 2	s2	Mendistribusikan lalu lintas dari s1 ke 2 *access switch* (s4 dan s5).
Switch 3	s3	Mendistribusikan lalu lintas dari s1 ke 2 *access switch* (s6 dan s7).
Switch 4	s4	Menghubungkan 7 *host* (h1-h7) ke jaringan. Titik awal akses pengguna ke sistem.
Switch 5	s5	Menghubungkan 7 *host* (h8-h14) ke jaringan. Titik awal akses pengguna ke sistem.
Switch 6	s6	Menghubungkan 8 *host* (h15-h22) ke jaringan. Titik awal akses pengguna ke sistem.
Switch 7	s7	Menghubungkan 8 *host* (h23-h30) ke jaringan. Titik awal akses pengguna ke sistem.
Host	h1-h30	*Node* atau titik akhir yang digunakan untuk mengirim/menerima data. Digunakan untuk simulasi serangan dan lalu lintas normal.

Tabel IV. 2 Perangkat dan Klasifikasi Host

IV.2.2 Implementasi Deteksi dan Mitigasi Serangan

Dalam penelitian ini, POX *Controller* digunakan sebagai pusat kendali jaringan sekaligus alat deteksi terhadap serangan ICMP *Flood*. *Controller* dirancang untuk membaca lalu lintas ICMP yang masuk, mengolah data paket, serta menghitung jumlah paket per detik yang dikirim oleh tiap alamat IP. Jika suatu IP menunjukkan lonjakan yang melebihi ambang batas tertentu, maka lalu lintas dari IP tersebut dicurigai sebagai serangan. Data yang dikumpulkan kemudian digunakan untuk melatih model klasifikasi menggunakan algoritma SVM. Model ini dilatih dengan data dari log serangan dan lalu lintas normal, lalu diujikan untuk melihat kemampuannya dalam membedakan keduanya. Hasil prediksi dari SVM disimpan dalam kolom khusus sebagai bagian dari laporan klasifikasi.

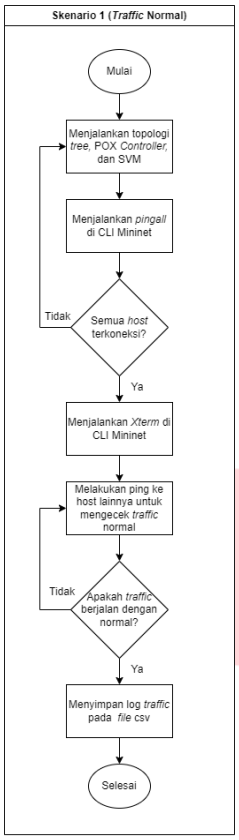
Setelah model berhasil memisahkan antara *traffic* normal dan serangan, tahap selanjutnya adalah mitigasi. Teknik mitigasi yang digunakan berbasis *rate limiting* dan dijalankan langsung oleh POX *Controller*. Ketika *controller* mendeteksi serangan yang berlangsung terus menerus selama 15 detik, maka sistem akan mengaktifkan blokir otomatis terhadap IP yang bersangkutan. Blokir ini dilakukan melalui pengiriman *flow rule* ke *switch* agar paket dari IP tersebut langsung di-*drop*, sehingga tidak mengganggu lalu lintas jaringan yang lain. Proses ini dijalankan secara otomatis dan dicatat dalam log untuk keperluan dokumentasi serta evaluasi.

Untuk mencatat seluruh aktivitas jaringan, *controller* telah diprogram untuk merekam data selama proses simulasi berjalan. Data dicatat dalam bentuk *file* icmpflood_log.csv yang berisi informasi lengkap tentang lalu lintas ICMP, seperti IP sumber dan tujuan, ukuran paket, jenis ICMP, serta waktu kejadian. *File* ini tidak hanya berguna untuk pelatihan model dan analisis, tapi juga menjadi dasar evaluasi efektivitas sistem deteksi dan mitigasi secara keseluruhan. Semua data log, termasuk hasil klasifikasi dan pemetaan deteksi, divisualisasikan dalam bentuk *confusion matrix* untuk menunjukkan performa sistem dalam menangani serangan dan membedakannya dari lalu lintas normal.

IV.3 Skenario Pengujian

Pengujian dilakukan untuk menilai kinerja deteksi dan mitigasi *ICMP Flood* pada jaringan *SDN* dengan *POX Controller*. Uji dilakukan dalam dua skenario: kondisi normal dan kondisi saat serangan terjadi. Proses ini mencakup pengecekan konektivitas, pengumpulan data, deteksi oleh *SVM*, dan aktivasi *rate limiting*.

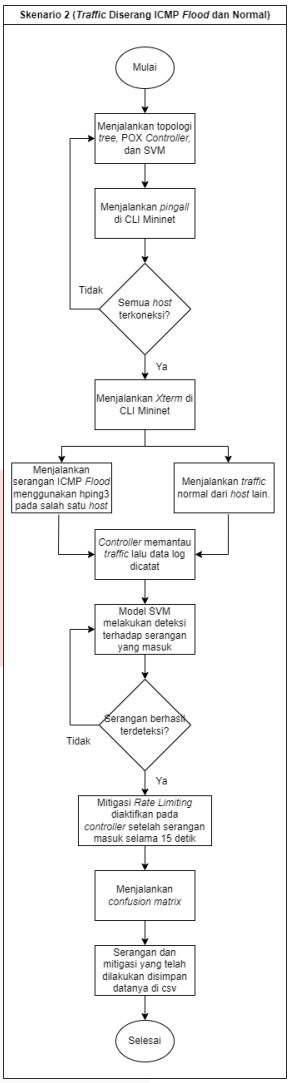
IV.3.1 Skenario 1 Pengujian pada *Traffic* Normal



Gambar IV. 2 *Flowchart* Skenario 1 *Traffic* Normal

Pada skenario 1, pengujian dilakukan dengan menjalankan topologi *tree*, POX *Controller*, dan model SVM. Seluruh host diuji konektivitasnya menggunakan perintah pingall di Mininet. Setelah semua host saling terhubung, beberapa host dibuka melalui xterm untuk melakukan *ping* ke host lain sebagai simulasi lalu lintas normal. Selama proses, aktivitas dicatat ke icmpflood_log.csv. Pada skenario ini tidak ada deteksi serangan maupun aktivasi mitigasi, sehingga lalu lintas yang tercatat hanya lalu lintas sah (*True Negative*).

IV.3.2 Skenario 2 Pengujian pada Traffic Diserang ICMP *Flood* bersamaan dengan *Traffic* Normal



Gambar IV. 3 *Flowchart* Skenario 2 *Traffic* ICMP *Flood* & Normal

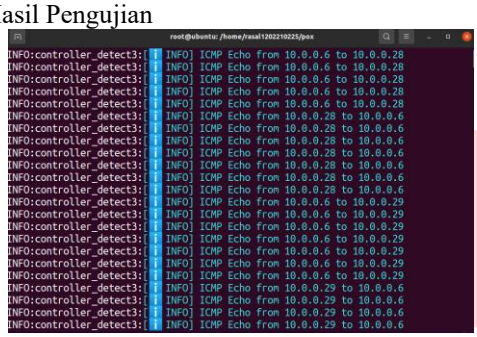
Pada skenario 2, pengujian dilakukan dengan menjalankan topologi *tree*, POX *Controller*, dan model SVM secara bersamaan. Seluruh *host* dicek konektivitasnya menggunakan pingall di Mininet. Jika koneksi lancar, beberapa *host* dibuka melalui Xterm untuk mensimulasikan lalu lintas normal sekaligus memicu serangan ICMP *Flood* menggunakan hping3 ke target dengan volume besar. *Host* normal tetap melakukan ping untuk memastikan lalu lintas sah tetap berjalan saat serangan aktif. Semua data ICMP, baik normal maupun *flood*, dicatat ke log. Model SVM memproses lalu lintas untuk memisahkan *traffic* normal dan serangan. Jika pola *flood* terdeteksi selama lebih dari 15 detik, *controller* otomatis mengaktifkan *rate limiting* untuk memblokir IP penyerang. Kinerja klasifikasi kemudian dievaluasi dengan *confusion matrix* dan hasilnya didokumentasikan. Uji dilakukan pada empat variasi: 3 dan 5 penyerang dengan intensitas tinggi dan sedang, untuk melihat pengaruh jumlah dan intensitas serangan terhadap performa *traffic* normal dan respon mitigasi sistem.

V. HASIL DAN ANALISIS

V.1 Skenario 1

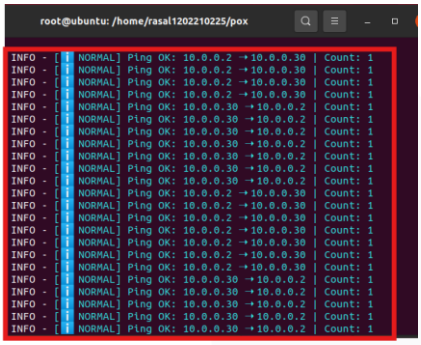
Pada skenario 1, pengujian difokuskan pada lalu lintas normal untuk memastikan sistem berjalan stabil tanpa salah klasifikasi saat tidak ada serangan. Topologi *tree* dan *POX Controller* dijalankan, lalu konektivitas host diperiksa dengan perintah *pingall* di Mininet. Selanjutnya, *ping* antar host diuji melalui *Xterm* untuk memastikan lalu lintas normal tetap lancar. Semua aktivitas lalu lintas pada skenario ini dicatat otomatis ke file log CSV.

V.1.1 Hasil Pengujian



Gambar V. 1 Tampilan *Controller* dijalankan

Gambar V.1 merupakan tampilan pada *controller* ketika *pingall* dilakukan.



Gambar V. 2 Tampilan SVM mendeteksi ping normal

Gambar V.2 merupakan tampilan dari model deteksi SVM ketika dijalankan pada traffic normal.

Gambar V. 3 Hasil Ping Normal di Xterm

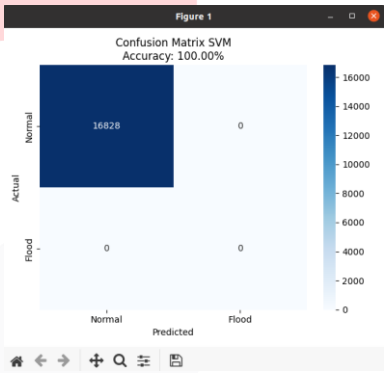
Gambar V-6 merupakan tampilan dari salah satu tes konektivitas pengujian di *node* Xterm dari h1 (10.0.0.1) ke h30 (10.0.0.30).

V.1.2 Analisis Pengujian

Parameter	h1 (10.0.0.1)	h2 (10.0.0.2)
Jumlah Paket Dikirim	31	301
Jumlah Paket Diterima	31	301
Packet Loss	0%	0%
Total Waktu Pengujian	30 detik	300 detik
RTT Minimum	8.643 ms	8.026 ms
RTT Rata-rata	11.988 ms	11.752 ms
RTT Maksimum	17.307 ms	21.368 ms
RTT *Mean Deviation*	2.002 ms	2.094 ms

Tabel V. 1 Hasil Uji Konektivitas Ping Normal

Tabel V.1 menampilkan hasil uji koneksi antar host sebagai dasar performa jaringan sebelum simulasi serangan dilakukan. Dalam uji ini, host h1 dan h2 mengirim *ping* ke h30 dengan durasi berbeda, yakni 30 detik untuk h1 dan 300 detik untuk h2, guna melihat kestabilan pada beban waktu pendek dan panjang. Hasilnya, semua paket terkirim tanpa *packet loss* (0%), dengan nilai RTT rata-rata 11–12 ms dan deviasi hanya sekitar 2 ms. RTT yang rendah dan deviasi kecil menandakan latensi stabil dan jalur komunikasi antar host berjalan lancar tanpa hambatan. Kondisi ini membuktikan topologi, switch, dan *controller* sudah berfungsi baik tanpa *bottleneck*, sehingga siap dipakai untuk uji deteksi dan mitigasi karena jalur data terbukti optimal pada kondisi normal.



Gambar V. 4 *Confusion Matrix Traffic* Normal

Gambar V.4 menampilkan *Confusion Matrix* untuk skenario lalu lintas normal dengan akurasi 100%. *True Negative* (16.828) tanpa muncul *False Positive*, *False Negative*, atau *True Positive*. Hal ini membuktikan bahwa model SVM mendeteksi lalu lintas normal dengan tepat tanpa menghasilkan alarm palsu. Tidak ada paket sah yang salah terbaca sebagai serangan, sesuai dengan kondisi uji yang memang tanpa serangan. Hasil ini menunjukkan sistem deteksi berjalan akurat di kondisi dasar, log ICMP tercatat sempurna, dan jalur komunikasi *host–controller* stabil tanpa gangguan. Dengan demikian, *baseline* jaringan dapat diandalkan untuk tahap uji serangan dan penerapan *rate limiting* selanjutnya.

V.2 Skenario 2

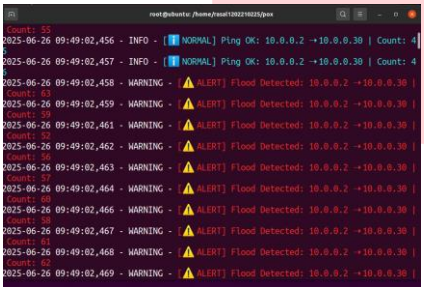
Pada skenario 2, pengujian dilakukan dengan memadukan empat variasi serangan dan lalu lintas normal untuk menilai kemampuan sistem membedakan traffic sah dan flood. Pengujian dimulai dengan menjalankan topologi *tree* dan *POX Controller*, lalu konektivitas host dicek melalui *pingall* di Mininet. Setelah semua host terhubung, *Xterm* digunakan untuk membuka node host normal dan host penyerang. Selanjutnya, *ping* normal dan serangan *ICMP Flood* dijalankan bersamaan. Sistem *SVM* mendeteksi jenis traffic dan jika flood terdeteksi selama 15 detik, *rate limiting* diaktifkan untuk memblokir IP penyerang. Semua data lalu lintas direkam dalam log CSV dan hasil klasifikasi ditampilkan melalui *confusion matrix* berisi *True Positive*,

True Negative, *False Positive*, dan *False Negative* untuk analisis lebih lanjut.

V.2.1 Hasil Pengujian

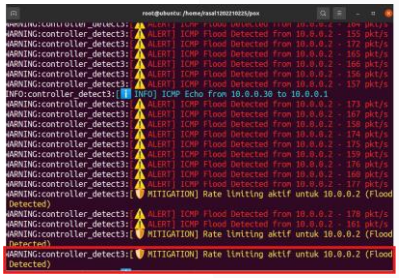
Gambar V. 5 Serangan ICMP *Flood* jenis pertama

Gambar V.5 menunjukkan bahwa salah satu *node* penyerang (h2) telah berhasil mengirim serangan ICMP ke *host* target (h30) dengan jumlah 100.000 paket dengan interval 1 ms per paket dan masing-masing paket memiliki 5000 *byte* data.



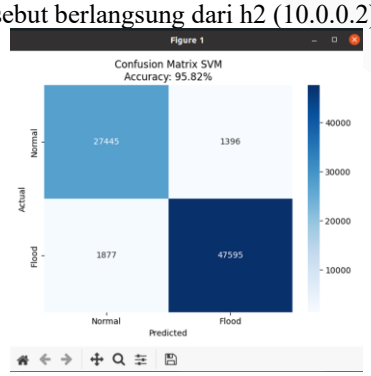
Gambar V. 6 Serangan yang terdeteksi pada SVM

Gambar V-9 merupakan tampilan dari deteksi SVM terhadap serangan ICMP yang masuk dari *host* penyerang (h2) ke *host* target (h30).



Gambar V. 7 Serangan jenis pertama berhasil dimitigasi

Gambar V-10 merupakan tampilan pada *controller* ketika serangan ICMP *Flood* dari salah satu *host* (h2) telah berhasil dimitigasi menggunakan *rate limiting* setelah selama 15 detik serangan tersebut berlangsung dari h2 (10.0.0.2).



Gambar V. 8 *Confusion Matrix* serangan jenis pertama

Gambar V.8 merupakan hasil *confusion matrix* dari serangan ICMP yang telah dilakukan oleh 3 *host* penyerang dengan serangan hping3 -1 -d 5000 -i u1000 -c 100000 10.0.0.30 dengan akurasi 95,82%.

V.2.2 Analisis Pengujian

Pengujian pada skenario serangan ICMP *Flood* yang dijalankan bersamaan dengan lalu lintas normal, dengan variasi jumlah penyerang dan tingkat intensitas, menghasilkan beberapa temuan berikut.

Jumlah Penyerang	Tingkat Serangan	Akurasi	*True Positive*	*False Positive*
2 Tanpa Mitigasi	hping3 -1 --flood	97,39%	1.982.563	23.134
3 dan *Traffic* Normal	hping3 -1 -d 5000 -i u1000 -c 100000 (Tinggi)	95,82%	47.595	1.396
	hping3 -1 -d 5000 -i u2500 -c 50000 (Sedang)	94,56%.	41.643	1.817
5 dan *Traffic* Normal	hping3 -1 -d 5000 -i u1000 -c 100000 (Tinggi)	93,48%.	42.196	2.589
	hping3 -1 -d 5000 -i u2500 -c 50000 (Sedang)	94,01%	43.645	2.156

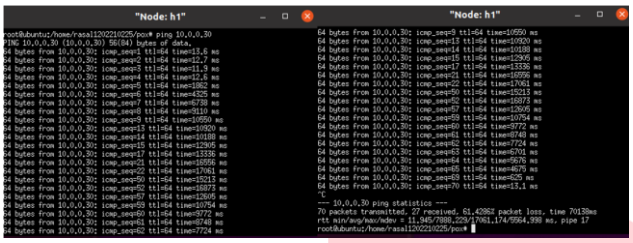
Tabel V. 2 Hasil Pengujuan Keempat Jenis Serangan

Berdasarkan Tabel V.2, akurasi deteksi SVM untuk lima skenario menunjukkan variasi kinerja yang signifikan. Pada skenario tanpa mitigasi, akurasi mencapai 97,39%, namun nilai FP melonjak hingga 23.134. Hal ini mengindikasikan bahwa meskipun sebagian besar serangan berhasil dikenali, masih banyak lalu lintas normal yang keliru terdeteksi sebagai ancaman, yang berpotensi mengganggu stabilitas jaringan karena paket sah ikut terblokir. Sebaliknya, pada empat skenario yang menerapkan mitigasi melalui *rate limiting*, akurasi sedikit lebih rendah di kisaran 93,48% hingga 95,82%, tetapi nilai FP jauh lebih kecil, yang menandakan peningkatan presisi dalam membedakan *traffic* normal dan serangan.

Skenario dengan tiga penyerang dan *traffic* normal berintensitas tinggi mencatat akurasi 95,82% dan FP yang relatif kecil, menunjukkan sistem dapat mengenali serangan secara efektif tanpa mengganggu trafik normal. Namun, saat intensitas serangan sedang, akurasi menurun menjadi 94,56% dan FP sedikit meningkat, kemungkinan karena pola *flood* menyerupai ping biasa sehingga sulit dibedakan. Pada skenario dengan lima penyerang dan intensitas tinggi, akurasi turun ke 93,48% dan FP meningkat, menunjukkan tantangan lebih besar dalam memilah traffic sah dari *flood*. Meski demikian, saat intensitas serangan diturunkan, akurasi meningkat menjadi 94,01% dan FP menurun, menandakan sistem mampu bekerja lebih baik dalam kondisi serangan yang tidak terlalu padat.

Secara umum, penerapan mitigasi memberikan dampak positif terhadap performa sistem. Walaupun akurasi sedikit berkurang dibanding skenario tanpa mitigasi, jumlah

kesalahan klasifikasi terhadap *traffic* normal menurun drastis. Hal ini menunjukkan bahwa mekanisme *rate limiting* tidak hanya membatasi serangan, tetapi juga membantu menjaga kestabilan jaringan dengan mengurangi potensi salah blokir terhadap paket sah. Temuan ini memperkuat bahwa integrasi deteksi SVM dan mitigasi pada POX *Controller* mampu memberikan keseimbangan antara keakuratan deteksi dan kelancaran lalu lintas normal di lingkungan SDN.



Gambar V. 9 *Traffic* Normal ketika *Host* lain menyerang

Gambar V.9 memperlihatkan aktivitas *host* normal yang mengirim ping ke IP 10.0.0.30 bersamaan dengan *host* penyerang ke alamat yang sama. Meskipun ada serangan ICMP *Flood*, *host* normal masih dapat mengirim dan menerima balasan meski tidak semua paket diterima: dari 70 paket hanya 27 yang sampai, dengan *packet loss* sekitar 61,42%. Rata-rata RTT tercatat naik hingga di atas 7 detik dengan deviasi lebih dari 5 detik, menandakan jalur kontrol terbebani. Meski begitu, sebagian paket tetap terkirim menunjukkan bahwa *rate limiting* di *controller* berfungsi membatasi dampak *flood*.

Parameter	2 Tanpa Mitigasi	3 + 1 Normal (Tinggi)	3 + 1 Normal (Sedang)	5 + 1 Normal (Tinggi)	5 + 1 Normal (Sedang)
Jumlah Paket Dikirim	1328	50	66	99	70
Jumlah Paket Diterima	981	14	36	39	27
Packet Loss Normal (%)	26,13	72	45,5	60,6	61,4
RTT Minimum (ms)	10,222	10,437	10,134	11,238	11,945
RTT Maksimum (ms)	115024,124	15622,664	19165,293	30757,328	17061,174
RTT Rata-rata (ms)	20861,078	4876,504	6553,869	11147,013	7888,229
RTT *Mean Deviation*	11502,601	5753,766	5133,652	8098,712	5564,998

Tabel V. 3 Rata-rata Metrik Ping Normal

Berdasarkan Tabel V.3, rata-rata pengujian ping normal pada lima skenario, termasuk satu skenario tanpa mitigasi, terlihat bahwa performa jaringan sangat dipengaruhi oleh keberadaan mitigasi. Pada skenario tanpa mitigasi, meskipun *packet loss* hanya sekitar 26%, RTT rata-rata mencapai 20.861 ms dengan deviasi hingga 11.502 ms, yang menunjukkan bahwa banyak paket memang terkirim, tetapi dalam kondisi keterlambatan yang ekstrem akibat beban serangan yang tidak di*filter*. Sebaliknya, skenario dengan mitigasi *rate limiting* memperlihatkan *packet loss* yang lebih tinggi, berkisar antara 45% hingga 72%, namun RTT rata-rata dan deviasi cenderung lebih terkendali. Misalnya, pada skenario dengan 3 penyerang berintensitas tinggi, *packet loss* tercatat 72%, namun RTT-nya masih di bawah 5.000 ms, dan deviasi sekitar 5.100 ms, yang berarti jalur komunikasi tetap terbuka meskipun ada antrean. Saat serangan dilakukan dengan intensitas sedang, angka *packet loss* menurun hingga 45%, dan RTT serta deviasinya menunjukkan stabilitas yang lebih baik. Pada skenario 5 penyerang dengan intensitas tinggi, RTT melonjak hingga di atas 11.000 ms dan deviasi mencapai 8.000 ms, namun angka *packet loss* sedikit lebih rendah dari skenario intensitas tertinggi dengan 3 penyerang, menunjukkan bahwa jumlah penyerang dan pola serangan sama-sama berpengaruh. Dari semua pengujian ini dapat disimpulkan bahwa mekanisme *rate limiting* cukup efektif dalam mengontrol beban serangan, karena masih ada sebagian lalu lintas normal yang bisa lewat, sementara keterlambatan bisa ditekan di bawah ambang ekstrem seperti yang terjadi saat tidak ada mitigasi. Meskipun tidak menghilangkan dampak sepenuhnya, mitigasi terbukti menjaga performa sistem tetap stabil dalam kondisi tertekan.

Skenario	Mulai *Delay* (icmp_seq)	Pulih Normal (icmp_seq)	Durasi Lonjakan (detik)
2 Penyerang Tanpa Mitigasi	5	1328	1323
3 Penyerang + 1 Normal (Tingkat Tinggi)	5	50	45
3 Penyerang + 1 Normal (Tingkat Sedang)	5	60	55
5 Penyerang + 1 Normal (Tingkat Tinggi)	5	99	94
5 Penyerang + 1 Normal (Tingkat Sedang)	5	70	65

Tabel V. 4 Perbandingan Durasi Lonjakan *Delay*

Hasil pengujian pada lima skenario ICMP *Flood*, baik dengan maupun tanpa mitigasi, menunjukkan bahwa kombinasi jumlah penyerang dan intensitas serangan sangat memengaruhi durasi gangguan pada jalur komunikasi normal. Tabel V.4 menunjukkan bahwa tanpa mitigasi, durasi lonjakan *delay* sangat tinggi, mencapai 1323 detik, akibat tidak adanya pembatasan lalu lintas *flood*. Sebaliknya, penerapan *rate limiting* terbukti mempercepat pemulihan jaringan. Misalnya, pada skenario tiga penyerang dengan intensitas tinggi, lonjakan hanya berlangsung 45 detik, sedangkan dengan intensitas sedang meningkat sedikit menjadi 55 detik, akibat pola serangan yang lebih menyerupai ping normal sehingga membutuhkan waktu

deteksi lebih lama. Ketika jumlah penyerang ditambah menjadi lima, durasi *delay* meningkat, yakni 94 detik untuk intensitas tinggi dan 65 detik untuk intensitas sedang. Lonjakan ini terjadi karena beban jaringan meningkat drastis akibat *flood* dari banyak sumber. Meski begitu, mitigasi tetap berhasil menjaga agar jalur tidak lumpuh total. Temuan ini memperkuat bahwa volume dan pola pengiriman paket sangat memengaruhi efektivitas deteksi, di mana serangan yang lebih padat justru lebih mudah dikenali oleh sistem. Namun, skenario dengan serangan menyebar dan lebih halus membutuhkan *threshold* deteksi dan strategi pemblokiran yang lebih adaptif. Dengan pengaturan yang lebih presisi, performa sistem dalam menjaga kestabilan lalu lintas sah dapat ditingkatkan, bahkan dalam kondisi serangan yang kompleks dan berskala besar.

VI. KESIMPULAN DAN SARAN

VI.1 Kesimpulan

Berdasarkan hasil penelitian, dapat disimpulkan bahwa serangan ICMP *Flood* pada jaringan SDN memunculkan pola lalu lintas yang khas yaitu pengiriman paket ICMP besar secara terus-menerus dalam jumlah besar, yang membebani POX *Controller* dan menyebabkan peningkatan *delay*, *packet loss*, hingga gangguan pada komunikasi normal. Tanpa mekanisme mitigasi, sistem kesulitan membedakan *traffic flood* dari lalu lintas sah, terlihat dari tingginya *nilai False Positive* (FP) mencapai 23.134 dan lonjakan *delay* hingga 1323 detik. Implementasi algoritma SVM terbukti efektif dalam mendeteksi serangan ini dengan akurasi stabil antara 93,48% hingga 95,82%, serta nilai *True Positive* (TP) yang tinggi, seperti 47.595 pada skenario tiga penyerang dengan intensitas tinggi. Setelah deteksi, *rate limiting* yang diterapkan di POX *Controller* mampu membatasi laju *flood* dengan cara memblokir IP sumber serangan, sehingga nilai FP menurun drastis ke kisaran 1.396–2.589 dan durasi gangguan berkurang menjadi 45–94 detik. Meskipun *packet loss* pada *traffic* normal masih cukup tinggi (45–72%), sistem tetap dapat menjaga sebagian lalu lintas sah tetap berjalan. Secara keseluruhan, integrasi SVM dan *rate limiting* terbukti menjadi pendekatan adaptif yang mampu menahan dampak serangan ICMP *Flood* dan meningkatkan kestabilan komunikasi pada arsitektur SDN.

VI.2 Saran

Sebagai tindak lanjut, beberapa saran dapat dipertimbangkan untuk pengembangan sistem ke depan. Penelitian selanjutnya disarankan mengatur parameter *threshold rate limiting* agar lebih adaptif terhadap berbagai skala serangan, sehingga *packet loss* pada lalu lintas normal bisa ditekan. Penggunaan algoritma *machine learning* lain atau kombinasi multi-model juga dapat meningkatkan akurasi deteksi. Uji coba di jaringan fisik atau lingkungan *cloud* perlu dilakukan agar hasil mendekati kondisi nyata. Selain itu, penambahan *dashboard* monitoring real-time akan membantu operator memantau serangan dan mitigasi secara visual dan responsif.

REFERENSI

[1] B. Al-Duwairi, E. Al-Quraan, and Y. AbdelQader, "ISDSDN: Mitigating SYN Flood Attacks in Software Defined Networks," *Journal of Network and Systems Management*, vol. 28, no. 4, pp. 1366–1390, Oct. 2020, doi: 10.1007/S10922-020-09540-1;WGROUP:STRING:ACM.

[2] Harshita, "Detection and Prevention of ICMP Flood DDOS Attack," 2017.

[3] J. F. . Kurose and K. W. . Ross, *Computer networking : a top-down approach*. Pearson, 2017.

[4] J. Ramprasath and V. Seethalakshmi, "Mitigation of Malicious Flooding in Software Defined Networks Using Dynamic Access Control List," *Wirel Pers Commun*, vol. 121, no. 1, pp. 107–125, Nov. 2021, doi: 10.1007/s11277-021-08626-6.

[5] C. Cortes, V. Vapnik, and L. Saitta, "Support-Vector Networks Editor," Kluwer Academic Publishers, 1995.

[6] C. M. Bishop, *Pattern Recognition and Machine Learning*. 2006.

[7] Andrew S. Tanenbaum and David J. Wetherall, *COMPUTER NETWORKS - A TANENBAUM - 5TH EDITION*, 5th ed. 2011.

[8] B. A. Forouzan, *Data Communications and Networking*, 4th ed. 2007.

[9] D. Kreutz, F. M. V. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," Oct. 2014, Accessed: Jun. 29, 2025. [Online]. Available: http://arxiv.org/abs/1406.0440

[10] W. Hill *et al.*, "DDoS in SDN: a review of open datasets, attack vectors and mitigation strategies," Sep. 01, 2024, *Springer Nature*. doi: 10.1007/s42452-024-06172-x.

[11] D. Prayoga, R. Muslim, and M. Husni, "Implementasi POX pada Perangkat Lunak Software-Defined Networking Controller untuk Data CENTER Berbasis Container," 2017.