

Intrusion Prevention System Using Random Forest and Rate limiting for Distributed Denial-of-Service

Diki Candra ^{1*}, Nanang Prihatin ^{2**}, Muhammad Davi ^{3*}

* Teknologi Informasi dan Komputer, Politeknik Negeri Lhokseumawe
dikisyandra@gmail.com ¹, nanang@pnl.ac.id ², muhammad.davi@pnl.ac.id ³

Article Info

Article history:

Received 2026-08-07

Revised 2026-09-11

Accepted 2026-10-15

Keyword:

*Distributed Denial of Service,
Intrusion Prevention System,
IPTables,
Security Server,
Random Forest,
Rate Limiting.*

ABSTRACT

Distributed Denial of Service (DDoS) attacks pose a major threat to web server availability, flooding servers with excessive traffic until Legitimate users can no longer reach the service. This study designs an Intrusion Prevention System (IPS) integrating Random Forest-based DDoS detection with iptables hashlimit rate limiting for automatic mitigation on an NGINX web server, reading traffic in real time from access logs rather than raw packet capture; its novelty lies in this end-to-end detection-mitigation integration rather than the individual algorithms. The model was trained on CIC-DDoS2019, mapped to eight deployment-consistent flow features across five classes: BENIGN, Syn, TFTP, DrDoS_UDP, and DrDoS_DNS. Random Forest reached 99.35% accuracy and 0.96 macro F1-score, outperformed Decision tree, SVM, and Logistic Regression baselines, and correctly flagged 99.67% of unseen attack samples as malicious, though BENIGN precision was only 69%. Rate limiting cut packet count by 99.65% (4,938.10 to 17.42 packets per interval) and CPU usage from 28.18% to 11.36%, with detection latency averaging 542.27 ms and iptables execution of just 3.84 ms. Legitimate-user testing showed 100% success rate and zero false positives when Attacker and Legitimate IPs differed, but revealed two limits of IP-based mitigation: shared-IP/NAT users suffered about 103 seconds of disruption from collateral blocking, and concurrent Legitimate traffic near the rate threshold triggered false positives in 5% of intervals. These results show Random Forest combined with iptables-based rate limiting mitigates DDoS effectively while exposing trade-offs between detection accuracy and service availability for real-world deployment.



This is an open access article under the [CC-BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.

I. PENDAHULUAN

Perkembangan layanan berbasis web pada sektor pendidikan, pemerintahan, industri, dan bisnis mendorong kebutuhan akan web server yang stabil menangani permintaan pengguna secara bersamaan. Nginx menjadi pilihan umum karena berfungsi sekaligus sebagai web server dan *reverse proxy* [1]. Peningkatan penggunaan layanan ini turut meningkatkan risiko serangan *Distributed Denial of Service* (DDoS), yang membanjiri target dengan trafik dalam jumlah besar hingga membebani sumber daya dan mengganggu ketersediaan layanan [2]. Dampaknya mencakup lonjakan penggunaan CPU, memori, dan *bandwidth*, serta terganggunya akses pengguna yang sah [2], [3].

Intrusion Detection System (IDS) berbasis *signature* dan *threshold* kesulitan mengenali pola serangan baru atau perubahan karakteristik trafik, sehingga rentan menghasilkan *false positive* dan *false negative* [4], [5]. *Machine learning* menutup celah ini dengan mempelajari pola trafik dari data historis. Faiz dkk. menekankan pentingnya rekayasa fitur dalam deteksi DDoS berbasis *Machine learning*, sedangkan Yuliswar dkk. menerapkan pendekatan serupa untuk mendeteksi serangan terdistribusi pada server [4], [6].

Random Forest menggabungkan banyak *decision tree* untuk menghasilkan klasifikasi yang lebih stabil. Kiswanto dkk. melaporkan akurasi 99,942% dalam deteksi DDoS menggunakan algoritma ini [5]. Fathurrahman dan Prabowo mencatat performa *Random Forest* yang mengungguli SVM

pada deteksi DDoS di jaringan *cloud* [7], dan Mansis dkk. menunjukkan hasil kompetitif dibandingkan K-Nearest Neighbor untuk klasifikasi serangan jaringan [8]. Temuan ini mendasari pemilihan *Random Forest* pada penelitian ini, meski performanya tetap perlu diuji terhadap karakteristik dataset dan lingkungan *deployment* yang digunakan.

Kesesuaian antara data pelatihan dan data *deployment* menjadi persoalan tersendiri dalam penerapan *Machine learning*. CIC-DDoS2019 menyediakan trafik benign dan berbagai kategori DDoS berbasis karakteristik *network flow* [2], sementara penelitian ini menerapkan model pada *access log Nginx* yang bersifat *real-time* dan berbasis aplikasi. Perbedaan *level* observasi ini berpotensi menimbulkan *distribution shift*. Sebagian fitur pada CIC-DDoS2019 sendiri, yang dihasilkan menggunakan CICFlowMeter, telah dilaporkan mengandung kesalahan perhitungan dan duplikasi pada penelitian sebelumnya [9], [10]. Penelitian ini menjawab kesenjangan tersebut dengan memetakan dan mengekstraksi delapan fitur dari *access log Nginx* agar strukturnya konsisten dengan masukan model: *pkt_count*, *byte_count*, *Request_rate*, *byte_rate*, *flow_duration*, *avg_packet_size*, *max_packet_size*, dan *min_packet_size*.

Penelitian ini menggunakan lima kelas CIC-DDoS2019: BENIGN, Syn, TFTP, DrDoS_UDP, dan DrDoS_DNS. Kelima kelas ini dipilih untuk membangun skenario klasifikasi sesuai kebutuhan penelitian, bukan untuk mewakili seluruh variasi DDoS di lingkungan nyata [2]. BENIGN merepresentasikan trafik normal, sedangkan empat kelas lainnya mewakili sebagian karakteristik serangan yang tersedia dalam dataset. Kemampuan model terhadap jenis serangan di luar *subset* ini tidak diklaim dalam penelitian ini.

Deteksi serangan memerlukan mekanisme mitigasi agar trafik berbahaya tidak terus membebani layanan. Dharma dan Muslikh menunjukkan bahwa *rate limiting* berbasis iptables mengurangi dampak HTTP GET Flood terhadap server [11]. IPS mengintegrasikan proses deteksi dengan tindakan pencegahan terhadap trafik yang teridentifikasi sebagai ancaman [12], [13]. Heremba dkk. menerapkan IPS berbasis Suricata untuk pengamanan dari serangan DDoS [14], sedangkan Alfarizi dkk. mengintegrasikan *Random Forest* dengan deteksi DDoS dari *log cPanel* disertai notifikasi *real-time* [15]. Penelitian lain turut menerapkan *Random Forest* untuk klasifikasi serangan berdasarkan data log dan karakteristik trafik [16], [17].

Penelitian sebelumnya membahas deteksi DDoS berbasis *Machine learning* dan mitigasi berbasis *rate limiting* secara terpisah. Integrasi keduanya pada web server dengan sumber data *access log Nginx* masih perlu dievaluasi dari sisi implementasi dan dampaknya terhadap kinerja server, terutama karena kesalahan klasifikasi trafik normal dapat memicu mitigasi terhadap pengguna yang sah. Penelitian ini karena itu tidak mengklaim kebaruan pada algoritma *Random Forest* atau iptables secara individual, melainkan pada integrasi keduanya dalam satu sistem IPS yang mendeteksi dan memitigasi secara otomatis.

Sistem yang dikembangkan memantau *access log Nginx* secara *real-time*, mengekstraksi delapan fitur trafik, lalu mengklasifikasikannya dengan *Random Forest*. Trafik yang terklasifikasi sebagai DDoS memicu penerusan *source IP* ke mekanisme *rate limiting* iptables *hashlimit*, lengkap dengan *penalty time* dan *penalty escalation* untuk sumber yang kembali terdeteksi menyerang. Performa klasifikasi dievaluasi melalui *accuracy*, *precision*, *recall*, F1-score, dan *confusion matrix*, sedangkan efektivitas mitigasi dianalisis dari perubahan *packet count*, *Requests/sec*, *throughput*, *delay*, CPU usage, dan RAM usage sebelum dan sesudah *rate limiting* diterapkan. Penurunan *Requests/sec* tidak serta-merta dibaca sebagai peningkatan kualitas layanan, karena pembatasan trafik berpotensi turut memengaruhi trafik pengguna yang sah.

Berdasarkan uraian tersebut, penelitian ini merancang dan mengimplementasikan *Intrusion Prevention System* yang mengintegrasikan deteksi DDoS berbasis *Random Forest* dengan mitigasi *rate limiting* berbasis iptables pada web server Nginx, dengan fokus evaluasi pada skenario pengujian yang mempertimbangkan perbedaan karakteristik data pelatihan dan *deployment* serta potensi dampak pembatasan trafik terhadap layanan.

II. METODE

Penelitian ini merancang dan mengimplementasikan *Intrusion Prevention System* (IPS) yang mendeteksi serta memitigasi serangan *Distributed Denial of Service* (DDoS) secara otomatis pada web server NGINX, dengan mengintegrasikan *Random Forest* untuk deteksi dan *rate limiting* berbasis iptables untuk mitigasi. *Random Forest* dipilih karena akurasi klasifikasinya tinggi terhadap lalu lintas jaringan, sedangkan *rate limiting* membatasi laju permintaan dari sumber serangan yang teridentifikasi secara sementara, bukan memblokirnya secara permanen.

A. Dataset CIC-DDoS2019

Penelitian ini menggunakan dataset CIC-DDoS2019 yang dikembangkan Canadian Institute for Cybersecurity, University of New Brunswick. Dataset ini menyediakan trafik benign dan berbagai kategori serangan DDoS berbasis *network flow*, mencakup SYN, UDP, TFTP, DNS, NTP, LDAP, MSSQL, NetBIOS, SNMP, SSDP, dan UDP-Lag [2]. Penelitian ini membatasi cakupan pada lima kelas: BENIGN, Syn, TFTP, DrDoS_UDP, dan DrDoS_DNS, disesuaikan dengan skenario pengujian dan fitur yang digunakan. BENIGN merepresentasikan trafik normal, sedangkan empat kelas lainnya merepresentasikan pola serangan dalam ruang lingkup penelitian, kelima kelas ini tidak mewakili seluruh jenis DDoS pada CIC-DDoS2019 maupun seluruh serangan yang mungkin terjadi pada web server NGINX [2].

Data melalui tahap pembersihan sebelum digunakan, baris dengan *missing value*, NaN, atau nilai *infinite* dihapus. Delapan fitur dipilih sebagai masukan model: *pkt_count*, *byte_count*, *Request_rate*, *byte_rate*, *flow_duration*, *avg_packet_size*, *max_packet_size*, dan *min_packet_size*,

untuk merepresentasikan volume, laju, durasi, dan ukuran trafik [4], [18]. Fitur-fitur ini bukan nama atribut asli CIC-DDoS2019, dataset menyediakan atribut *flow-level* seperti durasi aliran, jumlah paket, panjang paket, dan statistik ukuran paket [2], sehingga penelitian ini memetakan dan membentuk ulang fitur agar definisi dan strukturnya konsisten dengan data *real-time*. Detail pemetaan dijelaskan pada bagian pemetaan dan ekstraksi fitur trafik *real-time*.

Sebagian atribut asli yang menjadi rujukan penamaan fitur, seperti *Max Packet Length* dan *Min Packet Length*, dihitung secara *bidirectional* (mencakup arah *forward* dan *backward* pada satu *flow*), sedangkan definisi ulang fitur pada tahap *deployment* bersifat *forward-only* karena *access log Nginx* hanya mencatat permintaan dari sisi klien. Perbedaan cakupan observasi ini menjadi salah satu keterbatasan penyelarasan fitur antara data historis dan data *real-time*, dan perlu dipertimbangkan dalam interpretasi hasil klasifikasi pada tahap *deployment*.

CIC-DDoS2019 berfungsi sebagai data historis untuk membangun dan mengevaluasi model, sedangkan trafik pada tahap *deployment* diperoleh langsung dari *access log Nginx*; dataset tidak digunakan sebagai sumber trafik *real-time*. CIC-DDoS2019 merepresentasikan karakteristik *network flow*, sementara *access log Nginx* merepresentasikan aktivitas permintaan pada tingkat aplikasi, dan perbedaan *level* observasi ini dapat menimbulkan *distribution shift* antara data pelatihan dan data *deployment* [19]. Konsistensi definisi dan pembentukan fitur menjadi bagian penting untuk mengurangi dampak perbedaan ini pada implementasi model.

Setelah pembersihan dan pemilihan fitur, data difilter berdasarkan lima kelas yang ditentukan, lalu dibagi menjadi data latih dan data uji dengan rasio 80:20 menggunakan *stratified sampling* dan *random_state* 42. Data uji dipisahkan sebelum pelatihan dan tidak digunakan dalam pembentukan model. Tahapan pembagian data dan pelatihan *Random Forest* dijelaskan lebih lanjut pada bagian pelatihan model.

B. Pra-pemrosesan Data

Pra-pemrosesan menyiapkan data CIC-DDoS2019 dan data trafik *real-time* agar memiliki struktur masukan yang sama untuk model *Random Forest*, meliputi pembersihan data, pemilihan fitur dan kelas, pembagian data, serta penyelarasan struktur fitur antara data pelatihan dan *deployment*. Data CIC-DDoS2019 diperiksa terhadap *missing value*, NaN, dan nilai *infinite*; baris yang mengandung nilai tersebut dihapus sebelum digunakan dalam proses pembelajaran.

Delapan fitur digunakan sebagai masukan model: *pkt_count*, *byte_count*, *Request_rate*, *byte_rate*, *flow_duration*, *avg_packet_size*, *max_packet_size*, dan *min_packet_size*, merepresentasikan volume, laju, durasi, dan ukuran trafik. Rekayasa dan pemilihan fitur menjadi tahap penting dalam deteksi DDoS berbasis *Machine learning* karena karakteristik fitur memengaruhi kemampuan model membedakan trafik normal dan serangan [4]. Dataset kemudian difilter menjadi lima kelas: BENIGN merepresentasikan trafik normal, sedangkan Syn, TFTP, DrDoS_UDP, dan DrDoS_DNS merepresentasikan trafik

serangan dalam ruang lingkup penelitian [2], pembatasan ini bukan representasi seluruh jenis serangan pada CIC-DDoS2019.

Pada tahap *deployment*, data masukan diperoleh dari *access log Nginx*, memuat *source IP*, waktu akses, URL, dan ukuran data. *Request* pendukung seperti */favicon.ico*, */stream*, */health*, */metrics*, serta *file* statis (CSS, JavaScript, gambar, *font*) dikeluarkan agar tidak mendominasi karakteristik trafik yang diklasifikasikan. Data yang tersaring dikelompokkan berdasarkan *source IP* dan diagregasikan dalam interval 3 detik untuk membentuk delapan fitur dengan struktur yang sama seperti data pelatihan: *pkt_count* dari jumlah *Request*, *byte_count* dari total ukuran data, *Request_rate* dan *byte_rate* dari pembagian keduanya terhadap interval, *flow_duration* tetap 3 detik, serta *avg_packet_size*, *max_packet_size*, dan *min_packet_size* dari ukuran data dalam kelompok. Detail pembentukan fitur disajikan pada Tabel I.

TABEL I.
DEFINISI DAN PEMBENTUKAN FITUR *REAL-TIME*.

Fitur	Definisi dan Pembentukan
<i>pkt_count</i>	Jumlah <i>Request</i> yang tercatat dari satu <i>source IP</i> selama interval agregasi 3 detik.
<i>byte_count</i>	Total ukuran data dari seluruh <i>Request</i> yang tercatat dari satu <i>source IP</i> selama interval pengamatan.
<i>Request_rate</i>	Laju <i>Request</i> per detik yang dihitung dari <i>pkt_count</i> dibagi interval agregasi 3 detik.
<i>byte_rate</i>	Laju data per detik yang dihitung dari <i>byte_count</i> dibagi interval agregasi 3 detik.
<i>flow_duration</i>	Durasi interval pengamatan yang ditetapkan sebesar 3 detik.
<i>avg_packet_size</i>	Rata-rata ukuran data yang dihitung dari <i>byte_count</i> dibagi <i>pkt_count</i> .
<i>max_packet_size</i>	Ukuran data terbesar yang tercatat dari satu <i>source IP</i> selama interval pengamatan.
<i>min_packet_size</i>	Ukuran data terkecil yang tercatat dari satu <i>source IP</i> selama interval pengamatan.

Kelompok trafik dengan kurang dari 3 *Request* dalam satu interval tidak digunakan dalam klasifikasi untuk mengurangi data dengan aktivitas sangat rendah. Data hasil agregasi disimpan pada *traffic.csv* sebagai data trafik *real-time* dan diteruskan sebagai masukan model.

Pemilihan delapan fitur menggunakan definisi tetap sebelum data dibagi, sehingga tidak melibatkan seleksi fitur yang bergantung pada *label* atau distribusi data uji. Ketidakseimbangan kelas ditangani melalui parameter *class_weight="balanced"*, yang bekerja pada *level* algoritma selama pelatihan alih-alih mengubah komposisi data melalui *oversampling* atau *undersampling*. Data uji dengan demikian tetap terisolasi dari proses pembentukan model, sehingga risiko kebocoran informasi (*data leakage*) dari data uji ke tahap pelatihan dapat diminimalkan.

C. Pemetaan dan Ekstraksi Fitur Trafik *Real-time*

Pada tahap *deployment*, trafik aktual diperoleh dari *access log Nginx* yang memuat *source IP*, waktu akses, URL, dan

response size, kemudian diproses secara berkala untuk membentuk fitur masukan bagi model *Random Forest*. Pemanfaatan karakteristik trafik dan data log sebagai sumber informasi deteksi DDoS telah digunakan pada penelitian sebelumnya [4], [6], [15], [17].

Sebelum ekstraksi, sistem menyaring *Request* yang tidak merepresentasikan trafik utama pengujian */favicon.ico*, */stream*, */health*, */metrics*, serta *file* statis seperti CSS, JavaScript, gambar, dan *font* agar trafik pendukung aplikasi tidak mendominasi karakteristik trafik yang diklasifikasikan. *Request* yang tersaring dikelompokkan berdasarkan *source IP* dan diagregasikan dalam interval 3 detik untuk membentuk delapan fitur dengan struktur sama seperti tahap pelatihan. Pendekatan berbasis jumlah aktivitas, volume data, laju trafik, durasi, dan ukuran data juga digunakan dalam penelitian deteksi DDoS berbasis *Machine learning* lain [4], [5].

Pkt_count direpresentasikan sebagai jumlah *Request* atau entri *access log* dari satu *source IP* selama interval 3 detik, bukan jumlah paket jaringan dari *packet capture*. *Byte_count* dihitung dari total *response size*; *Request_rate* dan *byte_rate* masing-masing dari jumlah *Request* dan total *byte* dibagi interval 3 detik; *flow_duration* ditetapkan 3 detik sesuai interval agregasi; *avg_packet_size* dihitung dari *byte_count/pkt_count*; *max_packet_size* dan *min_packet_size* diperoleh dari nilai maksimum dan minimum *response size* pada kelompok trafik. Pemetaan sumber data dan pembentukan delapan fitur ditunjukkan pada Tabel II.

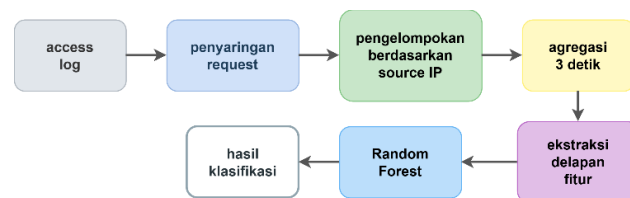
TABEL II.
PEMETAAN FITUR DENGAN INFORMASI ACCESS LOG NGINX.

Fitur Model	Sumber pada Access log Nginx	Cara Pembentukan pada Tahap Real-time
<i>pkt_count</i>	Jumlah <i>Request</i> dari satu <i>source IP</i>	Jumlah entri <i>Request</i> dalam interval 3 detik
<i>byte_count</i>	Ukuran data (<i>response size</i>) setiap <i>Request</i>	Penjumlahan ukuran data seluruh <i>Request</i>
<i>Request_rate</i>	Jumlah <i>Request</i> dan interval waktu	<i>pkt_count</i> /3 detik
<i>byte_rate</i>	Total ukuran data dan interval waktu	<i>byte_count</i> /3 detik
<i>flow_duration</i>	Interval agregasi trafik	Ditetapkan sebesar 3 detik
<i>avg_packet_size</i>	Ukuran data setiap <i>Request</i>	<i>byte_count</i> / <i>pkt_count</i>
<i>max_packet_size</i>	Ukuran data terbesar	Nilai maksimum <i>response size</i>
<i>min_packet_size</i>	Ukuran data terkecil	Nilai minimum <i>response size</i>

Setiap *source IP* menghasilkan satu vektor delapan fitur pada tiap interval 3 detik, dengan urutan dan struktur yang disamakan dengan masukan model pada tahap pelatihan. Penyelarasan ini diperlukan karena CIC-DDoS2019 berbasis *network flow*, sedangkan data *deployment* diperoleh dari aktivitas permintaan pada tingkat aplikasi melalui *access log*

Nginx; perbedaan sumber observasi ini dapat menyebabkan *distribution shift* dan memengaruhi kemampuan generalisasi model [2], [19].

Sistem hanya memproses kelompok dengan minimal 3 *Request* dalam satu interval untuk mengurangi data dengan aktivitas sangat rendah. Data yang memenuhi batas tersebut disimpan pada *traffic.csv* dan diteruskan sebagai masukan model *Random Forest*. Alur pembentukan fitur dari *access log* *Nginx* hingga menjadi masukan model ditunjukkan pada Gambar 1.



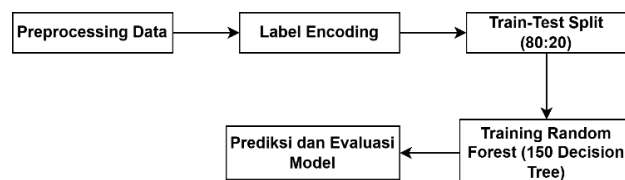
Gambar 1. Proses Ekstraksi Fitur.

D. Pelatihan Model Random Forest

Random Forest digunakan untuk membedakan trafik BENIGN dari empat kelas serangan (Syn, TFTP, DrDoS_UDP, DrDoS_DNS) dengan menggabungkan sejumlah *decision tree* untuk menghasilkan keputusan klasifikasi. Algoritma ini dipilih berdasarkan karakteristik data yang terdiri atas fitur numerik dan lima kelas trafik, serta hasil penelitian sebelumnya yang menunjukkan performa baik pada klasifikasi serangan jaringan [5], [7], [8], [17], [19].

Dataset hasil pra-pemrosesan dibagi menggunakan *stratified sampling* dengan rasio 80:20 dan *random_state*=42, mempertahankan proporsi setiap kelas pada kedua *subset*. Data uji dipisahkan sebelum pelatihan dan tidak digunakan dalam pembentukan model, sehingga tersedia untuk mengevaluasi prediksi terhadap data yang belum dipelajari model.

Model dikonfigurasi dengan 150 *decision tree*, *bootstrap*=True, dan *class_weight*="balanced" untuk mengatasi distribusi kelas yang tidak seimbang, sehingga kelas dengan jumlah sampel lebih sedikit tetap memperoleh bobot yang sesuai selama pembelajaran. Delapan fitur *pkt_count*, *byte_count*, *Request_rate*, *byte_rate*, *flow_duration*, *avg_packet_size*, *max_packet_size*, *min_packet_size* menjadi masukan model, dengan *label* trafik sebagai variabel target. Alur pelatihan, mulai dari dataset hasil pra-pemrosesan hingga penyimpanan model, ditunjukkan pada Gambar 2.



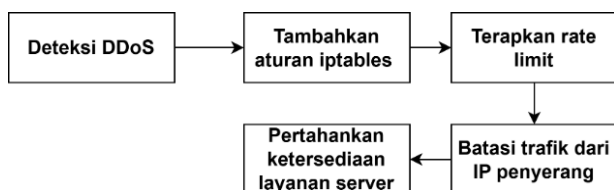
Gambar 2. Proses Training Model.

Setelah pelatihan, model disimpan dan digunakan pada tahap *deployment* untuk mengklasifikasikan fitur trafik dari *access log Nginx*; penyamaan struktur fitur antara data pelatihan dan *deployment* menjaga konsistensi definisi dan urutan atribut pada kedua tahap [4], [6]. Evaluasi menggunakan data uji yang telah dipisahkan sebelum pelatihan, diukur melalui *accuracy*, *precision*, *recall*, *F1-score*, dan *confusion matrix*, karena *accuracy* saja tidak cukup menggambarkan performa model pada dataset dengan distribusi kelas tidak seimbang, terutama untuk melihat kesalahan klasifikasi pada kelas BENIGN [5], [7], [8].

Model yang telah dilatih menjadi komponen deteksi pada sistem IPS: setiap vektor fitur hasil ekstraksi trafik *real-time* diklasifikasikan, dan apabila hasilnya termasuk kelas serangan, alamat IP sumber diteruskan ke mekanisme mitigasi *rate limiting* iptables.

E. Implementasi Rate limiting

Setelah *Random Forest* mengklasifikasikan trafik sebagai DDoS, sistem IPS menerapkan mitigasi menggunakan iptables dengan modul *hashlimit*, membatasi laju trafik berdasarkan alamat IP sumber tanpa melakukan pemblokiran permanen. Pendekatan ini juga diterapkan pada mitigasi HTTP GET Flood pada penelitian sebelumnya [11]. Alur implementasinya ditunjukkan pada Gambar 4.



Gambar 3. Proses Implementasi Rate limiting.

Mekanisme mitigasi menggunakan dua tingkat *penalty escalation* berdasarkan riwayat deteksi IP. Pada deteksi pertama, sistem menerapkan *rate limit* 10 paket/detik dengan *burst* 20 paket selama 5 menit, menggunakan *--hashlimit-mode srcip* pada trafik TCP menuju port 80, trafik yang melebihi batas dikenai DROP. Sistem memantau hasil klasifikasi setiap 3 detik dan memeriksa daftar *active_blocks* jika *rule* untuk IP tersebut masih aktif, tidak ada penambahan *rule* baru. Waktu deteksi dan waktu berakhirnya *penalty* disimpan melalui nilai *expiry*, dan *rule* dihapus otomatis begitu waktu tersebut tercapai, mengembalikan akses normal bagi IP yang bersangkutan.

Apabila IP yang sama kembali terdeteksi sebagai DDoS dalam 1 menit setelah *penalty* sebelumnya berakhir (diperiksa lewat *ratelimit_log.csv*), sistem menerapkan tingkat kedua: *rate limit* 5 paket/detik, *burst* 20 paket, *penalty* 10 menit. Deteksi ulang di luar rentang 1 menit tersebut diperlakukan sebagai deteksi baru dan kembali memperoleh tingkat pertama. Parameter lengkap dirangkum pada Tabel III.

TABEL III.

PARAMETER IMPLEMENTASI RATE LIMITING.

Parameter	Tingkat 1	Tingkat 2
<i>Rate limit</i>	10 paket/detik	5 paket/detik
<i>Burst</i>	20 paket	20 paket
<i>Mode hashlimit</i>	srcip	srcip
Protokol	TCP	TCP
<i>Destination port</i>	80	80
<i>Penalty time</i>	5 menit	10 menit
<i>Escalation window</i>	1 menit	-
Tindakan terhadap trafik di atas <i>limit</i>	DROP	DROP
Pemeriksaan prediksi	setiap 3 detik	setiap 3 detik

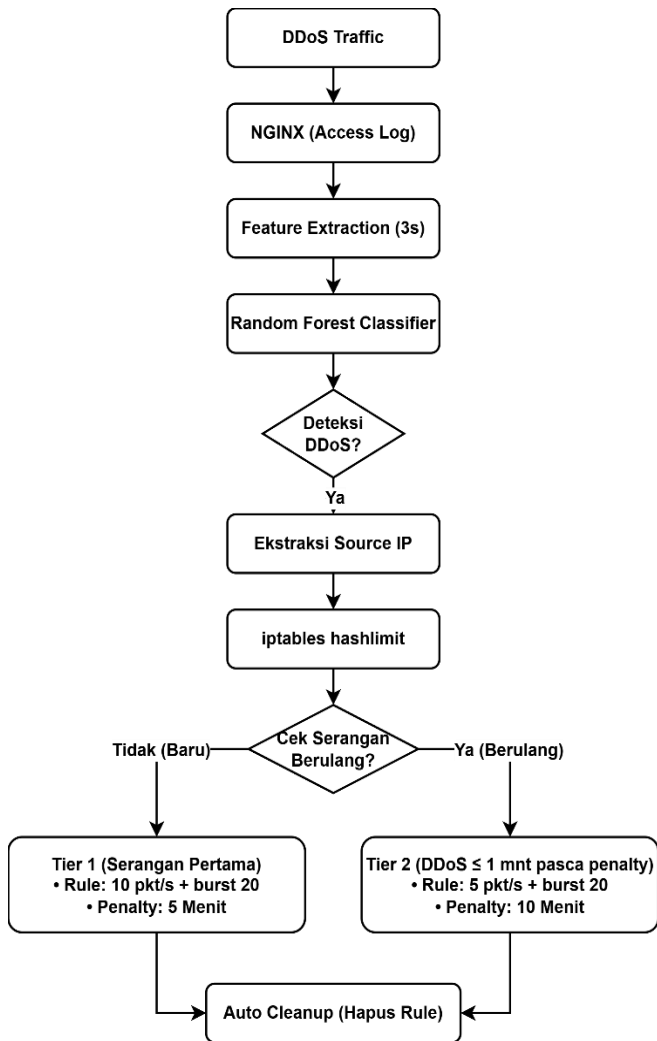
Konfigurasi ini bekerja dengan dua prinsip: pembatasan berdasarkan *source* IP dan peningkatan *penalty* berdasarkan riwayat deteksi, memungkinkan pembatasan individual sekaligus lebih ketat bagi IP yang berulang kali terdeteksi dalam *escalation window*. Penggunaan *source* IP sebagai dasar pembatasan tetap memiliki keterbatasan, IP bersama pada NAT/proxy dan serangan dengan banyak sumber atau perubahan IP dapat memengaruhi efektivitas mitigasi. Kondisi ini tidak diuji secara khusus dalam penelitian ini dan menjadi batasan ruang lingkup pengujian.

F. Arsitektur Sistem

Sistem IPS diimplementasikan pada VPS dengan web server NGINX, mengintegrasikan *monitoring* trafik, ekstraksi fitur, klasifikasi *Random Forest*, dan mitigasi iptables dalam satu alur, pencatatan trafik oleh NGINX, agregasi dan ekstraksi fitur, klasifikasi, lalu penerapan *rate limiting* terhadap trafik yang teridentifikasi sebagai serangan.

Trafik HTTP dicatat melalui *access log Nginx* dan diagregasi berdasarkan *source* IP dalam interval 3 detik untuk membentuk delapan fitur: *pkt_count*, *byte_count*, *Request_rate*, *byte_rate*, *flow_duration*, *avg_packet_size*, *max_packet_size*, dan *min_packet_size*, mengikuti pendekatan rekayasa fitur berbasis karakteristik trafik yang telah diterapkan pada penelitian sebelumnya [4], [6]. Data ini diberikan kepada model *Random Forest* yang telah dilatih menggunakan lima kelas CIC-DDoS2019 BENIGN, Syn, TFTP, DrDoS_UDP, DrDoS_DNS [2], [5] dengan struktur dan urutan fitur yang disesuaikan agar klasifikasi berjalan konsisten antara tahap pelatihan dan *deployment*.

Hasil klasifikasi menjadi dasar mitigasi *source* IP yang teridentifikasi sebagai trafik serangan dikenai *rate limiting* iptables *hashlimit*, membatasi trafik TCP menuju port 80 sebesar 10 paket/detik dengan *burst* 20 paket, sedangkan trafik yang melebihi batas dikenai DROP pendekatan yang juga diterapkan pada mitigasi HTTP GET Flood pada penelitian sebelumnya [11]. Proses deteksi dan mitigasi berjalan berkala setiap 3 detik, dengan *rule* mitigasi dikelola dan dihapus otomatis setelah masa *penalty* berakhir. Arsitektur system dapat dilihat pada gambar 4.



Gambar 4. Arsitektur sistem *Random Forest* dan *rate limiting*.

G. Skenario Pengujian

Pengujian dilakukan untuk mengevaluasi kemampuan IPS dalam mendeteksi dan memitigasi trafik DDoS pada web server NGINX di VPS, mencakup enam aspek kemampuan generalisasi model terhadap serangan baru, deteksi serangan, efektivitas *rate limiting*, kinerja *real-time* sistem, beban pada server, dan dampaknya terhadap pengguna sah.

Generalisasi terhadap *unseen attack* diuji menggunakan 300 sampel dari jenis serangan yang tidak termasuk dalam lima kelas pelatihan (DrDoS_LDAP, DrDoS_MSSQL, DrDoS_NTP, DrDoS_NetBIOS, DrDoS_SNMP, DrDoS_SSDP, LDAP, MSSQL, NetBIOS, UDP, dan UDP-lag), untuk menilai apakah model tetap dapat membedakan trafik serangan baru dari trafik BENIGN meskipun jenis serangannya belum pernah dipelajari.

Deteksi serangan dan efektivitas *rate limiting* diuji dengan mensimulasikan HTTP Flood menggunakan wrk (8 thread, 500 koneksi, durasi 5 menit) tanpa dan dengan *rate limiting* aktif, sementara sistem *monitoring* membaca *access log Nginx*

dan mengagregasikan trafik per *source IP* dalam interval 3 detik untuk klasifikasi *Random Forest*.

Kinerja *real-time* sistem diuji selama 100 detik dengan konfigurasi wrk yang sama (8 thread, 500 koneksi), mengukur *Prediction time*, *detection latency*, *mitigation latency*, dan waktu eksekusi iptables sejak trafik diterima hingga rule mitigasi diterapkan.

Kinerja server dievaluasi dengan membandingkan CPU usage, RAM usage, delay, throughput, dan Requests/sec antara kondisi tanpa dan dengan *rate limiting*, menggunakan beban pengujian yang sama seperti pengujian deteksi.

Ketersediaan layanan bagi *Legitimate user* diuji melalui lima skenario yang memvariasikan kondisi trafik pengguna sah bersama atau tanpa trafik serangan, dengan parameter yang diukur berupa *success rate*, *HTTP error rate*, dan *response time*:

1) Skenario A (Baseline Legitimate)

100 permintaan HTTP berurutan dengan jeda 1 detik tanpa trafik serangan, dijalankan menggunakan curl dari Kali Linux, mencatat waktu, kode status HTTP, dan *response time* setiap permintaan.

2) Skenario B (Legitimate dan Attacker, IP Berbeda)

Legitimate user menjalankan pengujian yang sama seperti Skenario A, bersamaan dengan *Attacker* yang menyerang menggunakan wrk dari *source IP* berbeda.

3) Skenario C (Legitimate dan Attacker, IP Sama)

Legitimate user menjalankan pengujian yang sama seperti Skenario A, bersamaan dengan *Attacker* yang menyerang menggunakan wrk dari *source IP* yang sama dengan *Legitimate user*, untuk mensimulasikan kondisi NAT atau *shared IP*.

4) Skenario D (Burst Traffic)

100 permintaan HTTP dikirim serentak secara paralel (bukan berurutan) dari satu *source IP* *Legitimate* menggunakan curl, dengan seluruh proses dijalankan bersamaan dan hasilnya dikumpulkan setelah semua permintaan selesai.

5) Skenario E (Concurrent Sustained Users)

10 pengguna paralel dari Kali Linux, masing-masing mengirim 30 permintaan HTTP dengan jeda 1 detik antar-permintaan, mensimulasikan akses bersamaan yang berkelanjutan dari beberapa pengguna sah.

H. Arsitektur Sistem

Evaluasi dilakukan berdasarkan hasil klasifikasi *Random Forest* menggunakan *accuracy*, *precision*, *recall*, *F1-score*, dan *confusion matrix*, sedangkan efektivitas mitigasi dibandingkan berdasarkan perubahan *pkt_count*, CPU, RAM, delay, throughput, dan Requests/sec. Penurunan throughput dan Requests/sec dianalisis sebagai konsekuensi pembatasan trafik, bukan indikator kegagalan, sedangkan efektivitas

sistem dinilai dari kemampuannya mengurangi trafik serangan dan beban server.

Pengujian dibatasi pada deteksi dan mitigasi berbasis *source* IP pada lingkungan VPS dan NGINX yang digunakan dalam penelitian. Pengujian tidak mencakup serangan terdistribusi dari banyak *source* IP secara bersamaan maupun skenario rotasi IP oleh *Attacker*, karena keterbatasan environment pengujian yang menggunakan satu IP penyerang. Dengan demikian, efektivitas mekanisme *rate limiting* berbasis *source* IP pada penelitian ini hanya tervalidasi untuk skenario serangan dari satu sumber, sedangkan ketahanannya terhadap serangan multi-sumber atau IP *rotation* belum dapat disimpulkan dan menjadi salah satu batasan utama ruang lingkup penelitian.

1) Evaluasi Kinerja Model Deteksi

Kinerja *Random Forest* dievaluasi menggunakan *accuracy*, *precision*, *recall*, *F1-score*, dan *confusion matrix*, karena distribusi jumlah data antarkelas tidak seimbang sehingga *accuracy* saja belum cukup menggambarkan kinerja setiap kelas [4], [5], [6], [7]. *Accuracy* menunjukkan proporsi prediksi benar terhadap seluruh data, *precision* menunjukkan ketepatan prediksi suatu kelas, *recall* menunjukkan kemampuan model mengenali data dari kelas sebenarnya, dan *F1-score* merupakan rata-rata harmonis *precision* dan *recall*. Persamaan *accuracy*, *precision*, *recall*, dan *F1-score* ditunjukkan berturut-turut pada Persamaan (1)-(4):

Persamaan *accuracy* ditunjukkan pada Persamaan (1).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \times 100\% \quad (1)$$

Persamaan *precision* ditunjukkan pada Persamaan (2).

$$Precision = \frac{TP}{TP+FP} \times 100\% \quad (2)$$

Persamaan *recall* ditunjukkan pada Persamaan (3).

$$Recall = \frac{TP}{TP+FN} \times 100\% \quad (3)$$

Persamaan *F1-score* ditunjukkan pada Persamaan (4).

$$F1-Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4)$$

Keterangan:

TP = True positive
TN = True Negative
FP = False positive
FN = False negative

Confusion matrix digunakan untuk melihat distribusi prediksi benar dan salah pada setiap kelas, karena kesalahan klasifikasi kelas BENIGN sebagai trafik serangan dapat menyebabkan alamat IP *Legitimate* memperoleh *rate limiting*. Oleh karena itu, hasil evaluasi tidak hanya dilihat dari *accuracy*, tetapi juga

precision, *recall*, *F1-score* setiap kelas, dan distribusi kesalahan pada *confusion matrix*.

2) Evaluasi Efektivitas Mitigasi

Efektivitas *rate limiting* dievaluasi dengan membandingkan *pkt_count*, *CPU usage*, *RAM usage*, *delay*, *throughput*, dan *Requests/sec* pada kondisi tanpa dan dengan *rate limiting*. Persentase perubahan setiap parameter dihitung menggunakan Persamaan (5):

Persentase perubahan dihitung menggunakan Persamaan (5).

$$Persentase P = \frac{NS - NSSD}{NS} \times 100\% \quad (5)$$

Keterangan:

Persentase P = Presentase Perubahan
NS = Nilai Sebelum
NSSD = Nilai Sesudah

Untuk parameter yang diharapkan menurun (*pkt_count*, *CPU usage*, *Requests/sec*), penurunan nilai menunjukkan berkurangnya trafik atau beban server. Peningkatan RAM tidak dianggap sebagai kegagalan mitigasi karena sistem IPS membutuhkan memori untuk klasifikasi, pencatatan data, pemantauan trafik, dan pengelolaan *rule iptables*, sedangkan penurunan *throughput* dan *Requests/sec* merupakan konsekuensi pembatasan trafik dan dianalisis bersama parameter lainnya, bukan berdiri sendiri.

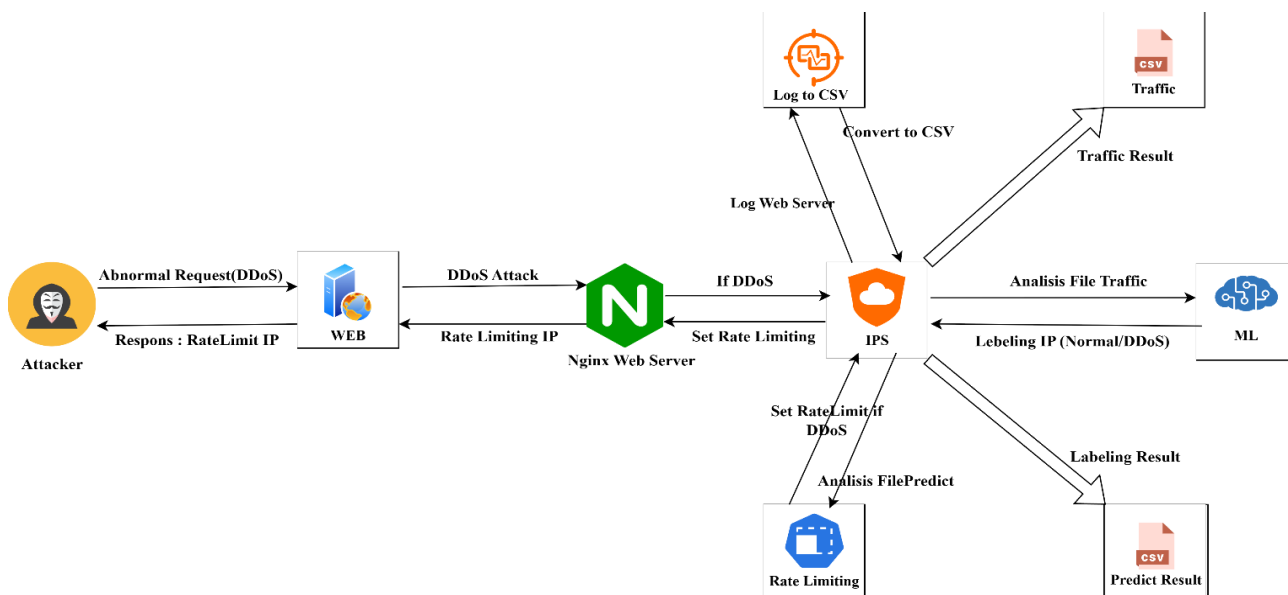
3) Evaluasi Kinerja Real-time Sistem

Selain akurasi klasifikasi dan efektivitas mitigasi, kecepatan respons sistem turut dievaluasi melalui *Prediction time*, *detection latency*, *mitigation latency*, dan waktu eksekusi *iptables*. Parameter ini penting karena akurasi tinggi tidak banyak berarti bagi sistem pencegahan intrusi apabila waktu antara trafik terdeteksi dan *rule* mitigasi diterapkan terlalu lama untuk mencegah dampak serangan secara efektif.

4) Evaluasi Ketersediaan Layanan bagi *Legitimate user*
Dampak mitigasi terhadap pengguna sah dievaluasi melalui *success rate*, *HTTP error rate*, dan *response time* pada berbagai kondisi trafik *Legitimate*, baik tanpa serangan, bersamaan dengan serangan dari IP berbeda maupun IP yang sama, maupun pada kondisi *burst* dan *concurrent access*. Parameter ini melengkapi evaluasi model dan mitigasi dengan menunjukkan dampak langsung yang dirasakan pengguna, bukan hanya dari sisi server.

5) Interpretasi Hasil Evaluasi

Hasil evaluasi model dan mitigasi dianalisis terpisah kemudian dihubungkan untuk menilai efektivitas keseluruhan sistem. Hasil klasifikasi *Random Forest* menjadi pemicu penerapan *rule iptables*, sehingga *false positive* memiliki konsekuensi operasional trafik *Legitimate* yang salah diklasifikasikan sebagai DDoS berpotensi memperoleh pembatasan. Oleh karena itu, *confusion matrix* dan metrik



Gambar 5. Rancangan Sistem.

klasifikasi digunakan untuk menilai risiko kesalahan sebelum hasil prediksi diterapkan pada mekanisme mitigasi otomatis. Efektivitas sistem secara keseluruhan dinilai dari keterkaitan antara kemampuan klasifikasi, perubahan *pkt_count*, CPU usage, RAM usage, delay, throughput, dan Requests/sec, kecepatan respons *real-time*, serta dampaknya terhadap ketersediaan layanan bagi pengguna sah bukan dari *accuracy* semata.

III. HASIL DAN PEMBAHASAN

Pembahasan mencakup implementasi sistem, pelatihan model, generalisasi terhadap serangan baru, deteksi dan mitigasi DDoS, kinerja *real-time*, beban CPU/RAM/delay, serta dampaknya terhadap pengguna sah. Hasil dianalisis untuk mengevaluasi efektivitas sistem menjaga ketersediaan layanan selama serangan berlangsung.

A. Implementasi Intrusion Prevention System (IPS)

Penelitian ini mengimplementasikan IPS berbasis *Machine learning* yang mengintegrasikan *Random Forest* untuk mendeteksi trafik DDoS dengan iptables *hashlimit* untuk mitigasi pada web server NGINX. Sistem menggunakan *access log Nginx* sebagai sumber data trafik HTTP yang diproses secara *real-time* untuk mengidentifikasi pola trafik dan menentukan tindakan mitigasi secara otomatis. *Random Forest* yang digunakan terdiri atas 150 *decision tree*, dipilih berdasarkan kemampuannya melakukan klasifikasi trafik jaringan dan mendeteksi pola serangan [13]. Implementasi IPS terdiri atas empat komponen NGINX, modul *monitoring*, modul prediksi *Random Forest*, dan modul mitigasi iptables *hashlimit*. Modul *monitoring* membaca *access log Nginx* dan mengagregasikan trafik berdasarkan *source IP* dalam interval 3 detik, membentuk delapan fitur masukan (*pkt_count*, *byte_count*, *Request_rate*, *byte_rate*, *flow_duration*, *avg_packet_size*, *max_packet_size*, *min_packet_size*) yang

kemudian diklasifikasikan *Random Forest* ke dalam lima kelas BENIGN, Syn, TFTP, DrDoS_UDP, dan DrDoS_DNS. Apabila hasil klasifikasi menunjukkan trafik serangan, *source IP* diteruskan ke modul mitigasi untuk menerapkan *rate limiting*. Rancangan integrasi kedua komponen ini ditunjukkan pada Gambar 5.

B. Implementasi dataset

Dataset yang digunakan adalah CIC-DDoS2019 yang dikembangkan Canadian Institute for Cybersecurity, berisi trafik normal dan berbagai jenis serangan DDoS. Sebelum digunakan untuk pelatihan *Random Forest*, dataset melalui *preprocessing* berupa pembersihan data, pemilihan fitur, dan penyaringan kelas sesuai ruang lingkup penelitian [1]. Baris yang mengandung *missing value*, NaN, dan *Infinity* dihapus, kemudian dipilih delapan fitur: *pkt_count*, *byte_count*, *Request_rate*, *byte_rate*, *flow_duration*, *avg_packet_size*, *max_packet_size*, dan *min_packet_size*. Dataset disaring menjadi lima kelas: BENIGN merepresentasikan trafik normal, sedangkan Syn, TFTP, DrDoS_UDP, dan DrDoS_DNS merepresentasikan trafik serangan [13].

Hasil *preprocessing* disimpan dalam *train_filtered.csv* yang terdiri atas delapan fitur masukan dan satu *label* klasifikasi, dengan contoh ditunjukkan pada Tabel IV, dan selanjutnya digunakan pada tahap pelatihan *Random Forest* untuk membangun model klasifikasi trafik pada sistem deteksi DDoS [15].

Sejumlah baris pada dataset hasil *preprocessing*, termasuk salah satu contoh pada Tabel IV, menunjukkan nilai *avg_packet_size* melebihi *max_packet_size*, kondisi yang secara matematis tidak seharusnya terjadi apabila kedua fitur dihitung pada cakupan observasi yang sama. CIC-DDoS2019 dihasilkan menggunakan tool CICFlowMeter-V3 yang sama dengan CIC-IDS2017 [2], [10].

TABEL IV.
DATASET CIC-DDoS2019 HASIL PREPROCESSING.

Pkt count	Byte count	Request rate	Byte rate	Flow duration	Avg packet size	Max packet size	Min packet size	Label
2	37.0	2000000.0	37000000.0	1	34.0	31.0	6.0	BENIGN
2	0	2000000	0	1	0	0	0	TFTP
2	766	2000000	766000000	1	574.5	383	383	DrDoS_UDP
2	1252	2000000	1252000000	1	939	626	626	DrDoS_DNS
8	48.0	0.26261151 07562261	1.5756690645 373566	38079062	6.6	6.0	6.0	Syn

Beberapa penelitian melaporkan kesalahan perhitungan dan duplikasi fitur pada *CICFlowMeter*, termasuk fitur terkait ukuran paket, dan menyatakan permasalahan tersebut berpotensi meluas ke seluruh dataset yang dihasilkan *tool* yang sama, termasuk CIC-DDoS2019 [9], [10]. Anomali ini kemungkinan besar merupakan artefak bawaan dataset sumber, bukan kesalahan pada tahapan *preprocessing* maupun pipeline penelitian ini, meskipun karakteristik pastinya pada CIC-DDoS2019 belum diaudit secara langsung sebagaimana dilakukan pada CIC-IDS2017.

Model menunjukkan performa tinggi pada seluruh kelas serangan DrDoS_DNS dan TFTP memperoleh *precision* 1,00 dan *F1-score* 1,00, sedangkan DrDoS_UDP dan Syn masing-masing mencapai *F1-score* 0,99. Kelas BENIGN, sebaliknya, memiliki *recall* 0,96 namun *precision* hanya 0,69. Berdasarkan *confusion matrix* pada Tabel VI, 92 dari 96 data BENIGN diklasifikasikan benar, sedangkan 34 data Syn, 6 data TFTP, dan 1 data DrDoS_DNS justru diprediksi sebagai BENIGN menunjukkan kemiripan karakteristik antara sebagian trafik normal dan trafik serangan pada fitur yang digunakan.

TABEL VI.
PERBANDINGAN RANDOM FOREST DENGAN BASELINE ALGORITMA LAIN.

Model	Accuracy	Precision	Recall	F1-Score	Waktu Training	Waktu Prediksi
Random Forest	0,9934	0,93	0,99	0,95	6,56 detik	0,3813 detik
Decision tree	0,9933	0,93	0,98	0,95	0,36 detik	0,0025 detik
SVM (linear)	0,9825	0,94	0,90	0,92	225,25 detik	13,0462 detik
Logistic Regression	0,9622	0,89	0,89	0,89	22,73 detik	0,0102 detik

C. Hasil Pelatihan Model Random Forest

Dataset hasil *preprocessing* dengan delapan fitur dan lima kelas (BENIGN, DrDoS_DNS, DrDoS_UDP, Syn, TFTP) digunakan untuk melatih *Random Forest*, dibagi 80:20 menggunakan *train-test split* dengan *stratify* untuk mempertahankan proporsi kelas. Model menggunakan 150 *decision tree* dengan *class_weight*=“*balanced*” untuk menangani ketidakseimbangan kelas, dan disimpan beserta *label encoder* serta daftar fitur dalam *random_forest.pkl* untuk tahap deteksi. Evaluasi terhadap 28.594 data uji menghasilkan *accuracy* 99,35%, dengan performa tiap kelas ditunjukkan pada Tabel V.

TABEL V.
CLASSIFICATION REPORT RANDOM FOREST.

Label	Precision	Recall	F1-Score	Support
BENIGN	0,69	0,96	0,80	96
DrDoS_DNS	1,00	0,99	1,00	4199
DrDoS_UDP	0,99	0,99	0,99	2610
Syn	0,98	0,99	0,99	5069
TFTP	1,00	0,99	1,00	16620
Accuracy	-	-	0,9935	28594
Macro Avg	0,93	0,99	0,96	28.594
Weighted Avg	0,99	0,99	0,99	28.594

TABEL VII.
CONFUSION MATRIX MODEL RANDOM FOREST.

Actual \ Predicted	BENIGN	DrDoS_DNS	DrDoS_UDP	Syn	TFTP
BENIGN	92	1	0	3	0
DrDoS_DNS	1	4.178	19	0	1
DrDoS_UDP	0	12	2.595	0	3
Syn	34	0	1	5.030	4
TFTP	6	3	0	99	16.512

Sebagian besar prediksi pada Tabel VII berada di diagonal utama kesalahan terbesar terjadi pada 99 data TFTP yang diprediksi sebagai Syn dan 34 data Syn yang diprediksi sebagai BENIGN, serta pertukaran prediksi antara DrDoS_DNS dan DrDoS_UDP (19 dan 12 data). Kesalahan ini relatif kecil dibandingkan total data tiap kelas. Secara keseluruhan, *accuracy* 99,35%, *macro F1-score* 0,96, dan *weighted F1-score* 0,99 mendukung penggunaan *Random Forest* sebagai komponen deteksi sebelum mitigasi *rate limiting* [17], meskipun *precision* BENIGN yang lebih rendah menunjukkan potensi kesalahan klasifikasi yang perlu diperhatikan karena hasil prediksi menjadi dasar mitigasi otomatis.

Untuk menguji signifikansi pemilihan *Random Forest*, dilakukan perbandingan dengan tiga *baseline Logistic Regression*, *Decision tree*, dan *SVM* kernel linear dilatih dan diuji menggunakan dataset, fitur, dan skema *train-test split* identik (80:20, *stratify*, *random_state*=42). Hasil perbandingan ditunjukkan pada Tabel VI.

Random Forest memperoleh *accuracy* dan *F1-score macro* tertinggi (0,9934 dan 0,95), serta *recall macro* 0,99 dan *recall* BENIGN 0,96, sedikit lebih baik dibandingkan *Decision tree*. *SVM* memiliki *precision macro* tertinggi (0,94), tetapi *recall macro* hanya 0,90 dan *recall* BENIGN 0,59, serta waktu komputasi lebih tinggi. *Logistic Regression* menunjukkan performa terendah pada sebagian besar metrik. Secara keseluruhan, *Random Forest* memberikan keseimbangan terbaik antara performa klasifikasi dan efisiensi komputasi untuk deteksi *real-time*.

D. Pengujian Generalisasi terhadap Unseen attack

Pengujian ini mengevaluasi kemampuan generalisasi *Random Forest* terhadap jenis serangan yang tidak digunakan saat pelatihan, menggunakan 300 sampel dari sebelas jenis serangan di luar lima kelas pelatihan (DrDoS_LDAP, DrDoS_MSSQL, DrDoS_NTP, DrDoS_NetBIOS, DrDoS_SNMP, DrDoS_SSDP, LDAP, MSSQL, NetBIOS, UDP, dan UDP-lag), sesuai skenario pengujian pada Bab II.

Karena *Random Forest* bersifat *closed-set* dan hanya mengenali lima kelas hasil pelatihan, serangan *unseen* tidak menghasilkan *label* khusus. Pengujian karena itu berfokus pada satu ukuran apakah model tetap membedakan trafik serangan baru dari BENIGN. Sampel dianggap berhasil dikenali jika diprediksi sebagai salah satu kelas serangan yang telah dikenal prediksi BENIGN dihitung sebagai *false negative*. Hasilnya ditunjukkan pada Tabel VIII.

TABEL VIII.
PREDIKSI PADA UNSEEN ATTACK.

Prediksi	Jumlah Sampel	Persentase
DrDoS_DNS	208	69,33%
DrDoS_UDP	85	28,33%
Syn	6	2,00%
BENIGN	1	0,33%
Total	300	100%

Sebanyak 299 dari 300 sampel (99,67%) terdeteksi sebagai serangan, sedangkan 1 sampel (0,33%) diklasifikasikan sebagai BENIGN. Model mampu membedakan trafik *unseen* dari trafik normal, tetapi belum dapat mengenali jenis serangan secara spesifik karena sebagian besar dipetakan ke DrDoS_DNS dan DrDoS_UDP. Oleh karena itu, nilai 99,67% merupakan *attack detection rate*, bukan akurasi klasifikasi jenis serangan baru. Rincian ditunjukkan pada Tabel IX.

Seluruh sampel DrDoS_LDAP, DrDoS_NetBIOS, DrDoS_SNMP, DrDoS_SSDP, LDAP, NetBIOS, dan UDP dikenali sebagai serangan meski terpetakan ke kelas berbeda dari *label* aslinya. Satu-satunya *false negative* terjadi pada DrDoS_NTP, dengan satu dari 39 sampel diklasifikasikan BENIGN.

TABEL IX.
KLASIFIKASI UNSEEN ATTACK.

Label Asli	BENIGN	DrDoS_DNS	DrDoS_UDP	Syn	Total
DrDoS_LDAP	0	23	0	0	23
DrDoS_MSSQL	0	31	7	0	38
DrDoS_NTP	1	37	0	1	39
DrDoS_NetBIOS	0	35	0	0	35
DrDoS_SNMP	0	44	0	0	44
DrDoS_SSDP	0	0	25	0	25
LDAP	0	12	0	0	12
MSSQL	0	30	11	1	42
NetBIOS	0	30	0	0	30
UDP	0	0	37	0	37
UDP-lag	0	0	2	1	3

Temuan ini menunjukkan fitur yang digunakan tetap mempertahankan kemampuan pemisahan trafik normal dari serangan meski jenisnya belum pernah dipelajari, sehingga model lebih tepat diposisikan sebagai mekanisme deteksi trafik mencurigakan daripada pengenalan jenis serangan spesifik. Implikasinya langsung terhadap IPS karena keputusan mitigasi hanya bergantung pada status serangan/BENIGN, 99,67% sampel *unseen* tetap berpotensi diteruskan ke mekanisme mitigasi, sementara satu sampel yang lolos sebagai BENIGN menegaskan risiko *false negative* tetap ada pada serangan yang benar-benar baru.

E. Hasil Deteksi Serangan DDoS

Pengujian deteksi disimulasikan menggunakan HTTP Flood dari wrk tanpa *rate limiting* (8 thread, 500 koneksi, durasi 5 menit), menghasilkan 493.317 *Request* dengan rata-rata 1.643,86 *Request*/detik, *throughput* 3,28 MB/s, dan *latency* rata-rata 234,74 ms. Skenario ini menghasilkan lonjakan trafik HTTP sebagai beban pengujian dan tidak dimaksudkan merepresentasikan botnet DDoS secara langsung.

Selama pengujian, sistem *monitoring* membaca *access log* Nginx dan mengagregasikan trafik per *source* IP dalam interval 3 detik untuk membentuk delapan fitur sesuai Bab II. *Source* IP 182.10.229.58 menunjukkan intensitas trafik tinggi dengan *pkt_count* 4.278-5.263 per interval, *Request_rate* 1.426-1.754,3 *Request*/detik, dan *byte_rate* 2.628.118-3.233.236 *byte*/detik, dengan *avg/max/min packet size* konsisten pada 1.843 *byte* mengindikasikan pola trafik seragam yang khas serangan otomatis. Hasil ekstraksi fitur ditunjukkan pada Tabel X.

TABEL X.
EKSTRAKSI FITUR LALU LINTAS SEBELUM *RATE LIMITING*.

<i>Timestamp</i>	<i>Source IP</i>	<i>Pkt count</i>	<i>Byte count</i>	<i>Request rate</i>	<i>Byte rate</i>	<i>Flow duration</i>	<i>Avg packet size</i>	<i>Max packet size</i>	<i>Min packet size</i>
7/6/2026 13:15:16	182.10.229.58	4278	7884354	1426	2628118	3	1843	1843	1843
7/6/2026 13:15:18	182.10.229.58	4802	8850086	1600.7	2950029	3	1843	1843	1843
7/6/2026 13:15:22	182.10.229.58	4888	9008584	1629.3	3002861	3	1843	1843	1843
7/6/2026 13:15:24	182.10.229.58	4391	8092613	1463.7	2697538	3	1843	1843	1843
7/6/2026 13:15:28	182.10.229.58	4746	8746878	1582	2915626	3	1843	1843	1843
7/6/2026 13:15:30	182.10.229.58	4874	8982782	1624.7	2994261	3	1843	1843	1843
7/6/2026 13:15:35	182.10.229.58	5086	9373498	1695.3	3124499	3	1843	1843	1843
7/6/2026 13:15:37	182.10.229.58	4903	9036229	1634.3	3012076	3	1843	1843	1843
7/6/2026 13:15:41	182.10.229.58	5263	9699709	1754.3	3233236	3	1843	1843	1843
7/6/2026 13:15:43	182.10.229.58	4870	8975410	1623.3	2991803	3	1843	1843	1843

TABEL XI.
KLASIFIKASI *TRAFFIC* SEBELUM *RATE LIMITING*.

<i>timestamp</i>	<i>source_ip</i>	<i>pkt_count</i>	<i>Prediction</i>
7/6/2026 13:15:16	182.10.229.58	4278	DDoS
7/6/2026 13:15:18	182.10.229.58	4802	DDoS
7/6/2026 13:15:22	182.10.229.58	4888	DDoS
7/6/2026 13:15:24	182.10.229.58	4391	DDoS
7/6/2026 13:15:28	182.10.229.58	4746	DDoS
7/6/2026 13:15:30	182.10.229.58	4874	DDoS
7/6/2026 13:15:35	182.10.229.58	5086	DDoS
7/6/2026 13:15:37	182.10.229.58	4903	DDoS
7/6/2026 13:15:41	182.10.229.58	5263	DDoS
7/6/2026 13:15:43	182.10.229.58	4870	DDoS

Kesepuluh vektor fitur pada Tabel X diteruskan ke model *Random Forest*, dengan hasil klasifikasi ditunjukkan pada Tabel XI. Seluruh sampel dari *source IP* tersebut diklasifikasikan DDoS. Nilai *pkt_count*, *Request_rate*, dan *byte_rate* yang tinggi serta konsisten menjadi karakteristik utama yang mendukung identifikasi ini, membuktikan bahwa fitur hasil transformasi dari *access log Nginx* tetap efektif digunakan sebagai masukan klasifikasi pada tahap *deployment*.

F. Hasil Implementasi *Rate limiting*

Pengujian dengan *rate limiting* menggunakan konfigurasi *wrk* yang sama seperti pengujian sebelumnya (8 *thread*, 500 koneksi, durasi 5 menit), agar perubahan trafik dapat dibandingkan langsung dengan kondisi tanpa mitigasi.

Iptables hashlimit berhasil membatasi laju trafik dari *source IP* yang teridentifikasi DDoS *Requests/sec* turun dari 1.643,86 menjadi 9,37 *Request/detik* (99,43%), *transfer/sec* turun dari 3,28 MB/s menjadi 19,13 KB/s (99,43%), sedangkan *latency* naik dari 234,74 ms menjadi 316,94 ms (35,00%) sebagai konsekuensi proses inspeksi dan mitigasi. Fitur hasil agregasi *access log Nginx* pada kondisi ini ditunjukkan pada Tabel XII, dengan hasil klasifikasinya pada Tabel XIII.

TABEL XII.
EKSTRAKSI FITUR LALU LINTAS SETELAH *RATE LIMITING*.

<i>Timestamp</i>	<i>Source IP</i>	<i>Pkt count</i>	<i>Byte count</i>	<i>Request rate</i>	<i>Byte rate</i>	<i>Flow duration</i>	<i>Avg packet size</i>	<i>Max packet size</i>	<i>Min packet size</i>
7/9/2026 7:32:34	182.10.229.58	20	36860	6.7	12286.7	3	1843	1843	1843
7/9/2026 7:32:39	182.10.229.58	21	38703	7	12901	3	1843	1843	1843
7/9/2026 7:32:41	182.10.229.58	23	42389	7.7	14129.7	3	1843	1843	1843
7/9/2026 7:32:45	182.10.229.58	23	42389	7.7	14129.7	3	1843	1843	1843
7/9/2026 7:32:47	182.10.229.58	23	42389	7.7	14129.7	3	1843	1843	1843
7/9/2026 7:32:51	182.10.229.58	23	42389	7.7	14129.7	3	1843	1843	1843
7/9/2026 7:32:53	182.10.229.58	20	36860	6.7	12286.7	3	1843	1843	1843
7/9/2026 7:32:55	182.10.229.58	24	44232	8	14744	3	1843	1843	1843
7/9/2026 7:32:59	182.10.229.58	21	38703	7	12901	3	1843	1843	1843
7/9/2026 7:33:01	182.10.229.58	25	46075	8.3	15358.3	3	1843	1843	1843

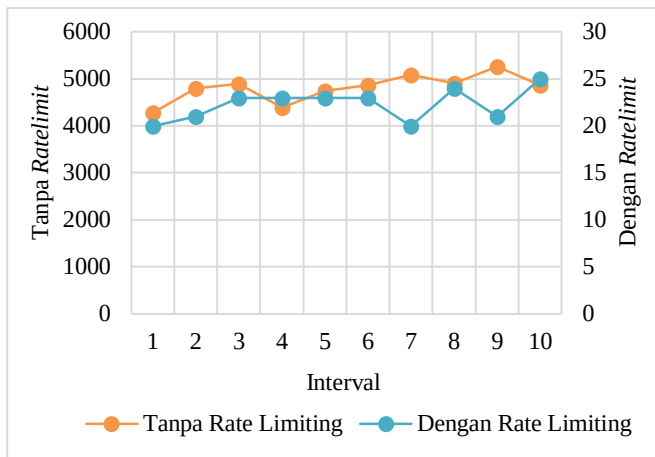
TABEL XIII.
KASIFIKASI *TRAFFIC* SETELAH *RATELIMITING*.

<i>timestamp</i>	<i>source_ip</i>	<i>pkt_count</i>	<i>Prediction</i>
7/9/2026 7:32:34	182.10.229.58	20	DDoS
7/9/2026 7:32:39	182.10.229.58	21	DDoS
7/9/2026 7:32:41	182.10.229.58	23	DDoS
7/9/2026 7:32:45	182.10.229.58	23	DDoS
7/9/2026 7:32:47	182.10.229.58	23	DDoS
7/9/2026 7:32:51	182.10.229.58	23	DDoS
7/9/2026 7:32:53	182.10.229.58	20	DDoS
7/9/2026 7:32:55	182.10.229.58	24	DDoS
7/9/2026 7:32:59	182.10.229.58	21	DDoS
7/9/2026 7:33:01	182.10.229.58	25	DDoS

Trafik dari *source IP* 182.10.229.58 tetap konsisten diklasifikasikan DDoS dengan *pkt_count* 20-25 per interval, membuktikan bahwa *rate limiting* tidak mengubah proses deteksi mitigasi diterapkan setelah trafik teridentifikasi sebagai serangan.

TABEL XIV.
SEBELUM DAN SESUDAH *RATE LIMITING*.

Kondisi	Jumlah Paket Tertinggi
Sebelum <i>Rate limiting</i>	4938.1
Sesudah <i>Rate limiting</i>	17.42
Persentase Penurunan	99.65%



Gambar 6. Perbandingan Jumlah Packet Count.

Berdasarkan Tabel XIV dan Gambar 6, rata-rata *pkt_count* menurun dari 4.938,10 menjadi 17,42 paket per interval (99,65%), menegaskan bahwa iptables *hashlimit* membatasi sebagian besar trafik serangan sebelum mencapai NGINX. Penurunan *Requests/sec* dan *throughput* yang menyertainya bukan indikator peningkatan kualitas layanan, melainkan konsekuensi langsung dari pembatasan trafik dampaknya terhadap CPU, RAM, *delay*, *throughput*, dan *Requests/sec* dianalisis lebih lanjut pada bagian evaluasi kinerja server.

G. Pengujian Kinerja Realtime Sistem IPS

Pengujian ini mengukur waktu yang dibutuhkan sistem untuk mendeteksi dan memitigasi trafik. Pengujian berlangsung 100 detik dengan konfigurasi wrk yang sama seperti sebelumnya (8 *thread*, 500 koneksi), mengamati *Prediction time*, *detection latency*, *mitigation latency*, dan waktu eksekusi iptables.

TABEL XV.
KINERJA REALTIME SISTEM IPS.

Parameter	Minimum	Median	Rata-rata	Maksimum
<i>Prediction time</i> (ms)	45,23	46,15	47,03	58,11
<i>Detection latency</i> (ms)	81,82	535,13	542,27	1.046,62

Berdasarkan Tabel XV, *Prediction time Random Forest* rata-rata 47,03 ms (45,23-58,11 ms) relatif konsisten sepanjang pengujian. *Detection latency end-to-end* jauh lebih tinggi, rata-rata 542,27 ms (median 535,13 ms, rentang 81,82-1.046,62 ms). Selisih ini menunjukkan waktu deteksi tidak hanya berasal dari klasifikasi *Random Forest*, tetapi juga mencakup waktu sistem membaca, memproses, dan meneruskan data trafik ke tahap klasifikasi.

TABEL XVI.
PENGUKURAN WAKTU DETEKSI DAN MITIGASI.

Parameter	Nilai
<i>Detection timestamp</i>	20:30:56.579245
<i>Mitigation timestamp</i>	20:30:57.588547
<i>Mitigation latency</i>	1.009,30 ms
<i>iptables execution time</i>	3,84 ms
<i>Rate limiting</i>	10 <i>Request/sec</i>
Durasi pembatasan	5 menit

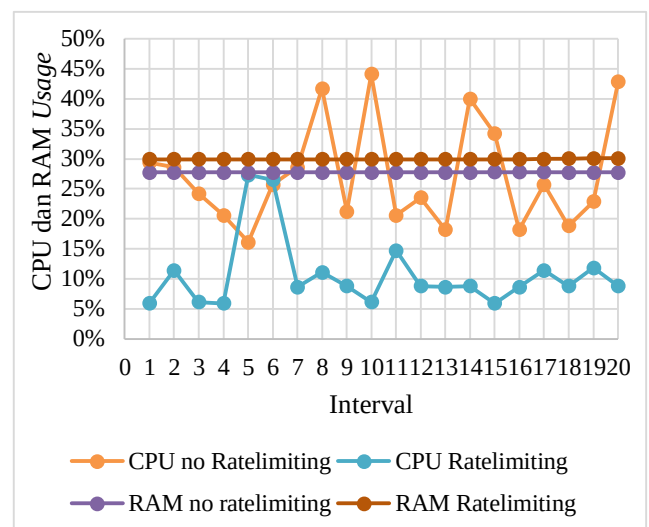
Rate limiting diterapkan 1.009,30 ms setelah deteksi, sedangkan eksekusi iptables hanya 3,84 ms, sehingga latensi didominasi tahap sebelum *rule* diterapkan. Seluruh 30 *record* trafik selama 100 detik pengujian terklasifikasi DDoS, menunjukkan *pipeline* ekstraksi-klasifikasi-mitigasi berjalan terintegrasi. Inferensi *Random Forest* rata-rata 47,03 ms jauh lebih cepat dari *detection latency end-to-end* 542,27 ms, menegaskan *bottleneck* ada pada alur pemrosesan, bukan model.

H. Kinerja Server (CPU, RAM, Throughput, Delay)

Evaluasi ini membandingkan kondisi tanpa dan dengan *rate limiting* untuk mengetahui pengaruh mitigasi terhadap penggunaan sumber daya dan karakteristik layanan web: CPU *usage*, RAM *usage*, *delay*, *throughput*, dan *Requests/sec*.

TABEL XVII.
CPU DAN RAM

Parameter	Tanpa <i>Rate limiting</i>	Dengan <i>Rate limiting</i>	Perubahan
CPU <i>Usage</i>	28.18%	11.36%	-59,67%
RAM <i>Usage</i>	27.36%	29.59%	+8,16%

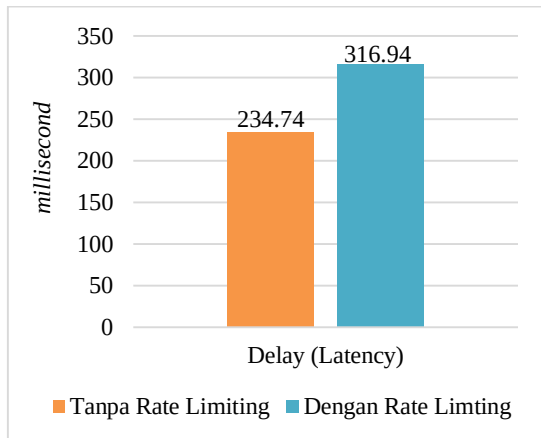


Gambar 7. Perbandingan penggunaan CPU dan RAM

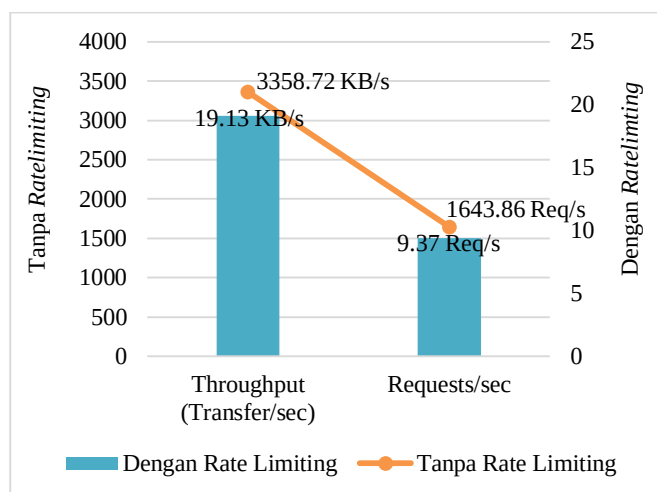
Berdasarkan Tabel XVII dan Gambar 7, CPU *usage* turun 59,67% karena pembatasan trafik mengurangi jumlah permintaan yang diproses web server, sedangkan RAM *usage* naik 8,16% akibat *overhead* sistem IPS pemantauan, pencatatan log, agregasi trafik, ekstraksi fitur, klasifikasi *Random Forest*, dan pengelolaan *rule iptables*. Penerapan IPS dengan demikian menghasilkan *trade-off*: beban CPU berkurang signifikan, sementara kebutuhan memori meningkat meski relatif kecil.

TABEL XVIII.
DELAY, THROUGHPUT, DAN REQUESTS/SEC.

Parameter	Tanpa Rate limiting	Dengan Rate limiting	Perubahan
Delay (Latency)	234,74 ms	316,94 ms	+35,00%
Throughput (Transfer/sec)	3358.72 KB/s	19,13 KB/s	-99,43%
Requests/sec	1.643,86 req/s	9,37 req/s	-99,43%



Gambar 8. Delay sebelum dan sesudah rate limiting



Gambar 9. Throughput dan Requests/sec

Berdasarkan Tabel XVIII dan Gambar 8-9, *Requests/sec* dan *throughput* turun 99,43%, menunjukkan sebagian besar trafik pengujian berhasil dibatasi sehingga volume yang diteruskan ke web server jauh lebih rendah penurunan ini merupakan konsekuensi pembatasan trafik, bukan indikator penurunan kualitas layanan. *Delay* di sisi lain naik 35,00% akibat tambahan waktu pemeriksaan dan pembatasan trafik sebelum permintaan diteruskan ke layanan web kenaikan ini terjadi bersamaan dengan penurunan volume trafik yang jauh lebih besar.

Secara keseluruhan, *rate limiting* menurunkan CPU *usage* (59,67%), *Requests/sec* (99,43%), dan *throughput* (99,43%), dengan konsekuensi kenaikan RAM *usage* (8,16%) dan *delay* (35,00%). Mitigasi efektif mengurangi volume trafik dan beban pemrosesan server, dengan *overhead* tambahan pada memori dan waktu respons layanan.

I. Pengujian Ketersediaan Layanan bagi Legitimate user

Bagian D, E, dan F mengevaluasi dampak *rate limiting* terhadap trafik serangan dan kinerja server, tetapi belum menunjukkan dampaknya bagi pengguna sah secara langsung. (A) *baseline* tanpa serangan, (B) *Legitimate* dan *Attacker* pada IP berbeda, (C) *Legitimate* dan *Attacker* pada IP sama, (D) *Burst Traffic*, (E) *Concurrent Sustained Users* dengan parameter *success rate*, *HTTP error rate*, dan *response time*.

1) Skenario A - Baseline Legitimate

TABEL XIX.
SKENARIO A (BASELINE).

Parameter	Nilai
Success Rate	100%
HTTP Error Rate	0%
Response Time rata-rata	77,25 ms
Response Time median	76,43 ms
Std deviasi	2,85 ms
Response Time max	86,90 ms

Berdasarkan Tabel XIX, Seluruh permintaan berhasil diproses dengan *response time* stabil (72-87 ms, std deviasi 2,85 ms), menjadi acuan bagi skenario berikutnya.

2) Skenario B - Legitimate user dan Attacker pada Source IP Berbeda

TABEL XX.
SKENARIO B (IP BERBEDA).

Parameter	Nilai
Success Rate	100%
HTTP Error Rate	0%
Response Time rata-rata	81,86 ms
Response Time median	75,24 ms
Std deviasi	35,81 ms
Response Time max	346,20 ms

Berdasarkan Tabel XX, *Response time* median (75,24 ms) tetap sebanding dengan Skenario A meski rata-rata dan std deviasi naik akibat dua permintaan dengan lonjakan di atas 300 ms. *Log* klasifikasi IPS menunjukkan *source IP Legitimate* tercatat BENIGN pada seluruh 27 interval (0% *false positive*), sementara *source IP Attacker* konsisten terklasifikasi DDoS dengan *pkt_count* turun dari ribuan menjadi 15-20 paket per interval setelah *rate limiting* aktif mitigasi berbasis IP tidak berdampak pada *Legitimate user* selama IP keduanya berbeda.

- 3) Skenario C - *Legitimate user* dan *Attacker* pada *Source IP* yang Sama

TABEL XXI.
SKENARIO C (SHARED IP).

Parameter	Nilai
Success Rate (HTTP 200)	100%
Response Time rata-rata	1.110,15 ms
Response Time median	76,42 ms
Std deviasi	7.623,30 ms
Response Time max	68.686,56 ms
Permintaan dengan <i>response time</i> >1 detik	2 dari 100 (2%)
Durasi gangguan layanan	±103 detik

Berdasarkan Tabel XXI, Meski seluruh permintaan akhirnya memperoleh HTTP 200, *success rate* 100% tidak mencerminkan pengalaman sebenarnya satu permintaan tertahan 34,03 detik dan permintaan berikutnya 68,69 detik sebelum trafik kembali normal, sehingga *source IP* ini mengalami gangguan layanan ±103 detik. Pola *response time* yang naik bertahap mengindikasikan *retransmisi* TCP akibat DROP oleh iptables *hashlimit* karena mitigasi bekerja berdasarkan *source IP* tanpa membedakan jenis trafik, *Legitimate user* yang berbagi IP dengan *Attacker* turut terdampak meski bukan bagian dari serangan.

- 4) Skenario D - *Burst Traffic* dari *Legitimate user*.

TABEL XXII.
SKENARIO D (BURST TRAFFIC).

Parameter	Nilai
Total permintaan	100
Success Rate	100%
Response Time rata-rata (87 permintaan awal)	93,81 ms
Response Time rata-rata (13 permintaan akhir)	881,15 ms
Response Time max	2.113,61 ms
<i>pkt_count</i> per interval	93 dan 7
Request rate tertinggi	31,0 Request/detik
Prediction	Normal (kedua interval)

Berdasarkan Tabel XXII, Seluruh 100 permintaan berhasil diproses, tetapi 13 permintaan terakhir mengalami peningkatan waktu tunggu bertahap hingga 2.113,61 ms pola *queueing* akibat keterbatasan kapasitas koneksi simultan, bukan *rate limiting*, karena klasifikasi tetap Normal pada kedua interval meski *Request rate* (31,0/detik) tiga kali melampaui ambang Level 1 (10 paket/detik). Model tidak salah mengenali *burst Legitimate* sebagai DDoS, tetapi kapasitas server tetap menjadi faktor terpisah yang membatasi performa saat volume tinggi datang serentak.

- 5) Skenario E - *Concurrent Sustained Users*

TABEL XXIII.
SKENARIO E (CONCURRENT SUSTAINED USERS).

Parameter	Nilai
Jumlah pengguna paralel	10
Total permintaan	300
Durasi pengujian	119,87 detik
Success Rate	92,33%
Error Rate	7,67%
Response Time rata-rata	1.663,73 ms
Response Time median	1.074,41 ms
Response Time min	72,36 ms
Response Time max	13.433,39 ms
Response Time p95	6.234,70 ms
Total interval pengamatan	40 interval
Interval diklasifikasikan DDoS	2 dari 40 (5,00%)
Volume trafik dalam interval salah klasifikasi	54 dari 278 paket (19,42%)

Kesalahan klasifikasi terjadi pada dua interval awal (*pkt_count* 29 dan 25 *Request rate* 9,7 dan 8,3/detik) tepat saat kesepuluh *thread* mulai mengirim permintaan bersamaan; interval berikutnya dengan *pkt_count* di bawah 10 konsisten Normal. *Response time* median yang tinggi (1.074,41 ms) dan *error rate* 7,67% menunjukkan dampak dua interval yang salah klasifikasi ini menjalar ke permintaan-permintaan berikutnya akibat *rate limiting* berlangsung selama periode penalti berbeda dari Skenario A, B, dan D, di sini *false positive* murni berasal dari trafik *Legitimate* tanpa keberadaan serangan, menandakan ambang batas deteksi perlu disesuaikan untuk trafik dengan intensitas sedang yang meningkat mendadak.

- 6) Perbandingan Kelima Skenario Perbandingan hasil kelima skenario ditunjukkan pada Tabel XXIV.

TABEL XXIV.
SKENARIO C (SHARED IP).

Parameter	A	B	C	D	E
<i>Success Rate</i>	100%	100%	100%	100%	92,33%
<i>Response Time median</i>	76,43 ms	75,24 ms	76,42 ms	93,16 ms	1.074,41 ms
<i>Response Time max</i>	86,90 ms	346,20 ms	68.686,56 ms	2.113,61 ms	13.433,39 ms
<i>False positive pada Legitimate IP</i>	Tidak ada	Tidak ada	Tidak ada	Tidak ada	5,00% (2/40 interval)
Gangguan layanan signifikan	Tidak ada	Tidak ada	±103 detik	Antrian sesaat (13 <i>Request</i> akhir)	Kegagalan <i>Request</i> (7,67%) akibat <i>false positive</i>

Response time median pada Skenario A, B, dan C konsisten di 75-76 ms sistem tidak membebani *Legitimate user* pada trafik sekuensial normal, dengan atau tanpa serangan bersamaan. Skenario D sedikit lebih tinggi (93,16 ms) akibat antrian sesaat tanpa kesalahan klasifikasi, sedangkan Skenario E melonjak signifikan (1.074,41 ms, lebih dari 14 kali *baseline*) akibat kesalahan klasifikasi pada fase awal.

Perbedaan paling signifikan ada pada Skenario C vs E. Skenario C menunjukkan *collateral damage* saat *Legitimate user* berbagi IP dengan *Attacker* sungguhan, sedangkan Skenario E menunjukkan *false positive* dapat muncul murni dari trafik *Legitimate* tanpa serangan sama sekali, saat volume dari beberapa pengguna bersamaan mendekati ambang *rate limiting* (5,00% interval salah klasifikasi, berdampak pada kegagalan 7,67% permintaan). Kedua temuan menunjukkan efektivitas mitigasi berbasis IP bergantung pada dua faktor: asumsi satu IP mewakili satu pengguna (Skenario C), dan ketepatan ambang batas deteksi terhadap trafik *Legitimate* yang meningkat mendadak (Skenario E) keduanya perlu jadi pertimbangan penyesuaian *threshold* pada pengembangan lanjutan.

J. Pembahasan

Model *Random Forest* mendeteksi trafik DDoS dengan akurasi 99,35% serta *precision*, *recall*, dan *F1-score* tinggi pada sebagian besar kelas serangan, menunjukkan kemampuan membedakan trafik normal dan serangan berdasarkan delapan fitur yang digunakan sejalan dengan penelitian sebelumnya [13], [17]. Perbandingan dengan *baseline* algoritma lain menunjukkan *Random Forest* unggul pada *accuracy* dan *recall macro* dibanding *Logistic Regression*, *Decision tree*, dan *SVM*, terutama pada kelas minoritas BENIGN, dengan waktu komputasi yang jauh lebih efisien dibanding *SVM* memperkuat justifikasi pemilihan algoritma ini untuk kebutuhan klasifikasi *real-time*.

Pengujian generalisasi terhadap *unseen attack* menunjukkan 99,67% dari 300 sampel serangan di luar lima kelas pelatihan tetap dikenali sebagai trafik serangan, meski model tidak mampu mengidentifikasi jenis serangannya secara spesifik.

Temuan ini menegaskan model lebih tepat diposisikan sebagai mekanisme deteksi trafik mencurigakan daripada pengenalan jenis serangan baru, sekaligus menunjukkan risiko *false negative* tetap ada meski kecil.

Pada tahap mitigasi, *rate limiting* berbasis *iptables hashlimit* mengurangi volume trafik secara signifikan: rata-rata *packet count* turun 99,65% (4.938,10 menjadi 17,42 paket per interval), diikuti penurunan *Requests/sec* dan *throughput* masing-masing 99,43%. Pengujian kinerja *real-time* menunjukkan *Prediction time Random Forest* stabil (rata-rata 47,03 ms), sedangkan *detection latency end-to-end* jauh lebih tinggi (rata-rata 542,27 ms) karena turut mencakup waktu pembacaan dan pemrosesan trafik, dengan *mitigation latency* sekitar 1,01 detik dan eksekusi *rule iptables* yang sangat singkat (3,84 ms) menunjukkan sebagian besar latensi berasal dari alur pemrosesan sistem, bukan dari *iptables* itu sendiri.

Mitigasi turut memengaruhi sumber daya dan waktu respons server CPU *usage* turun 59,67% seiring berkurangnya beban pemrosesan, sedangkan RAM *usage* naik 8,16% akibat *overhead monitoring*, ekstraksi fitur, klasifikasi, pencatatan log, dan pengelolaan *rule iptables*, dengan *delay* naik 35,00% sebagai konsekuensi proses pembatasan trafik.

Pengujian terhadap *Legitimate user* melengkapi temuan di atas dengan menunjukkan dampak mitigasi dari sisi pengguna. Selama *source IP Legitimate user* berbeda dari *Attacker* (Skenario A dan B), *success rate* tetap 100% dengan *response time* sebanding *baseline* dan 0% *false positive*. Namun, dua kondisi mengungkap batasan mitigasi berbasis IP: pada Skenario C, *Legitimate user* yang berbagi IP dengan *Attacker* sungguhan (kondisi NAT/shared IP) mengalami gangguan layanan ±103 detik akibat *collateral damage* dari DROP oleh *iptables* pada Skenario E, kesalahan klasifikasi justru muncul murni dari trafik *Legitimate* tanpa keberadaan serangan sama sekali, ketika volume dari beberapa pengguna bersamaan (*Concurrent Sustained Users*) mendekati ambang batas *rate limiting*, menyebabkan 5,00% interval salah diklasifikasikan sebagai DDoS dan berdampak pada kegagalan 7,67% permintaan berikutnya. Skenario D menunjukkan model tetap tidak salah mengklasifikasikan *Burst Traffic* instan sebagai serangan meski *Request rate* sempat tiga kali melampaui

ambang batas, meski kapasitas server tetap membatasi performa saat volume tinggi datang serentak.

Secara keseluruhan, hasil penelitian menunjukkan *trade-off* antara keamanan dan performa layanan integrasi *Random Forest* dan iptables *hashlimit* efektif mendeteksi dan membatasi DDoS, termasuk terhadap serangan yang tidak pernah dipelajari, dengan kecepatan respons yang memadai untuk kebutuhan *real-time*. Namun, efektivitas mitigasi berbasis *source IP* bergantung pada dua faktor yang saling terpisah asumsi satu IP mewakili satu entitas pengguna, dan ketepatan ambang batas deteksi terhadap volume trafik *Legitimate* yang wajar namun meningkat mendadak sehingga penilaian keberhasilan sistem IPS tidak dapat hanya didasarkan pada akurasi deteksi maupun efektivitas pengurangan trafik, melainkan juga pada dampaknya terhadap ketersediaan layanan bagi pengguna sah [15], [17].

IV. KESIMPULAN

Random Forest berhasil diimplementasikan sebagai komponen deteksi pada sistem *Intrusion Prevention System* (IPS) untuk mengidentifikasi serangan DDoS. Model dengan delapan fitur dan 150 *decision tree* memperoleh akurasi 99,35% serta *precision*, *recall*, dan *F1-score* tinggi pada sebagian besar kelas serangan [13], [17]. Perbandingan dengan *baseline* algoritma lain menunjukkan *Random Forest* unggul tipis dari *Decision tree* (*accuracy* 99,34% berbanding 99,33%) dan jauh melampaui SVM serta *Logistic Regression*, terutama pada *recall* kelas minoritas BENIGN (0,96), dengan waktu komputasi jauh lebih efisien dibanding SVM (0,38 detik berbanding 13,05 detik). Pengujian generalisasi terhadap *unseen attack* menunjukkan 99,67% dari 300 sampel serangan di luar kelas pelatihan tetap dikenali sebagai trafik serangan, meski model tidak mengidentifikasi jenis serangannya secara spesifik.

Mekanisme *rate limiting* berbasis iptables *hashlimit* menurunkan rata-rata *packet count* dari 4.938,10 menjadi 17,42 paket per interval (99,65%), *Requests/sec* dari 1.643,86 menjadi 9,37 *Request/detik*, dan *CPU usage* dari 28,18% menjadi 11,36%, dengan konsekuensi *RAM usage* naik menjadi 29,59% dan *delay* naik menjadi 316,94 ms. Pengujian kinerja *real-time* menunjukkan *Prediction time Random Forest* stabil (rata-rata 47,03 ms), sedangkan *detection latency end-to-end* lebih tinggi (rata-rata 542,27 ms) karena turut mencakup alur pemrosesan sistem, dengan eksekusi *rule iptables* yang sangat singkat (3,84 ms).

Pengujian terhadap *Legitimate user* menunjukkan mitigasi tidak mengorbankan ketersediaan layanan selama *source IP Legitimate* berbeda dari *Attacker* (Skenario A, B, D): *success rate* tetap 100% dengan *response time* sebanding *baseline* dan 0% *false positive*, termasuk pada kondisi *Burst Traffic* instan. Namun, dua kondisi mengungkap batasan mitigasi berbasis IP. Pada Skenario C (*shared IP/NAT*), *Legitimate user* yang berbagi IP dengan *Attacker* mengalami gangguan layanan ± 103 detik akibat *collateral damage* dari DROP oleh iptables. Pada Skenario E (*Concurrent Sustained Users*), kesalahan klasifikasi justru muncul murni dari trafik *Legitimate* tanpa

serangan sama sekali ketika volume dari beberapa pengguna bersamaan mendekati ambang batas *rate limiting*, menyebabkan 5,00% interval salah diklasifikasikan sebagai DDoS dan berdampak pada kegagalan 7,67% permintaan berikutnya.

Secara keseluruhan, integrasi *Random Forest* dan iptables *hashlimit* mampu mendeteksi dan memitigasi DDoS secara terintegrasi dan responsif, termasuk terhadap serangan yang tidak pernah dipelajari, dengan *trade-off* berupa tambahan penggunaan RAM dan *delay*. Efektivitas mitigasi berbasis IP bergantung pada dua faktor asumsi satu IP mewakili satu entitas pengguna, dan ketepatan ambang batas deteksi terhadap volume trafik *Legitimate* yang meningkat mendadak keduanya menjadi celah yang berpotensi menimbulkan *collateral damage* maupun *false positive*.

Penelitian ini memiliki keterbatasan yang perlu ditindaklanjuti. Pertama, *robustness* terhadap serangan DDoS terdistribusi dari banyak *source IP* maupun *IP rotation* belum diuji. Kedua, evaluasi pada kondisi *shared IP/NAT* berskala lebih besar dan penyesuaian *threshold rate limiting* untuk mengurangi *false positive* pada trafik *concurrent Legitimate* masih perlu diperluas. Penelitian lanjutan disarankan menguji sistem dengan sumber trafik dari beberapa titik jaringan berbeda secara bersamaan serta mengevaluasi penyesuaian ambang batas deteksi, agar efektivitas dan batasan mitigasi berbasis IP dapat dipahami lebih menyeluruh.

DAFTAR PUSTAKA

- [1] P. Satya Saputra, P. Aditya Pratama, and L. Putu Ary Sri Tjahyanti, "Perancangan Dan Komparasi Web Server Nginx Dengan Web Server Apache Serta Pemanfaatan Reverse Proxy Server Pada Nginx," *Jurnal Komputer dan Teknologi Sains (KOMTEKS)*, vol. 2, no. 1, pp. 16–21, 2023.
- [2] A. A. G. Iman Sharafaldin, Arash Habibi Lashkari, Saqib Hakak, "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy," *2019 International Carnahan Conference on Security Technology (ICCSST), Chennai, India*, no. Cic, 2019, doi: 10.1109/CCST.2019.8888419.
- [3] W. M. A. Dimas Satrya Bhayangkara, "Pengaruh Load Balancing Pada Serangan DDoS Menggunakan Nginx," *The Indonesian Journal of Computer Science*, vol. 13, no. 4, 2024, [Online]. Available: <https://doi.org/10.33022/ijcs.v13i4.4118>
- [4] Muhammad Nur Faiz, Oman Somantri, and Arif Wirawan Muhammad, "Rekayasa Fitur Berbasis Machine Learning untuk Mendeteksi Serangan DDoS," *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*, vol. 11, no. 3, pp. 176–182, 2022, doi: 10.22146/jnteti.v11i3.3423.
- [5] D. Kiswanto, F. Ramadhani, N. M. Surbakti, N. A. Nasution, F. Matematika, and I. P. Alam, "Pengembangan dan Implementasi Sistem Deteksi Serangan DDoS Berbasis Algoritma Random Forest," *Bulletin of Information Technology (BIT)*, vol. 6, no. 3, pp. 247–256, 2025, doi: 10.47065/bit.v6i3.2203.
- [6] T. Yuliswar, I. Elfriti, and O. W purbo, "Optimization of Intrusion Detection System with Machine Learning for Detecting Distributed Attacks on Server," *INOVTEK Polbeng - Seri Informatika*, vol. 10, no. 1, pp. 367–376, 2025, doi: 10.35314/vem9da98.
- [7] D. W. P. Muhammad Andika Fathurrahman, "Perbandingan Performa Algoritma Random Forest dan SVM Dalam Mendeteksi serangan DDoS di Jaringan Cloud Comparison of the Performance of Random Forest and SVM Algorithms in Detecting DDoS Attacks in Cloud Networks," *Jurnal Riset Rekayasa Elektro*, vol. 7, no. 2, pp. 193–202, 2025, doi: DOI : 10.30595/jrre.v7i2.27866.

- [8] M. I. Mansis, F. S. Putri, and M. R. Siregar, "Jurnal Processor Analisis Kinerja Algoritma K-Nearest Neighbor dan Random Forest," *Jurnal Processor*, vol. 20, no. 2, pp. 134–145, 2025, doi: <https://doi.org/10.33998/processor.2025.20.2.2594>.
- [9] A. Rosay, F. Carlier, E. Cheval, and P. Leroux, "From CIC-IDS2017 to LYCOS-IDS2017: A corrected dataset for better performance," *ACM International Conference Proceeding Series*, pp. 570–575, 2021, doi: 10.1145/3486622.3493973.
- [10] G. Engelen, V. Rimmer, and W. Joosen, "Troubleshooting an Intrusion Detection Dataset: The CICIDS2017 Case Study," *Proceedings - 2021 IEEE Symposium on Security and Privacy Workshops, SPW 2021*, pp. 7–12, 2021, doi: 10.1109/SPW53761.2021.00009.
- [11] B. S. Dharma and A. R. Muslikh, "Implementation of rate limiting and Telegram bot for HTTP GET Flood attack mitigation," vol. 3, no. 1, pp. 58–67, 2025, doi: 10.26905/jisad.v3i1.15398.
- [12] T. Rahman and I. Rozen, "Strategi Penguatan Keamanan Jaringan dengan IDS dan IPS di PT. Toppan Plasindo Lestari Cibitung," *Ijccs*, vol. x, No.x, no. x, pp. 1–5, 2024.
- [13] A. Rahmatillah, A. Firdaus, and E. Laila, "Implementasi Intrusion Prevention System (IPS) Pada Keamanan Jaringan Dengan Notifikasi Berbasis Telegram di Jurusan Teknik Komputer," *Jurnal Laporan Akhir Teknik Komputer*, vol. 1, no. 1, pp. 10–17, 2021.
- [14] I. Heremba, N. S. Irjanto, and T. B. Bun, "Implementasi Intrusion Prevention System Menggunakan Suricata Sebagai Pengamanan Dari Serangan DDoS," *Progresif: Jurnal Ilmiah Komputer*, vol. 21, no. 1, p. 364, 2025, doi: 10.35889/progresif.v21i1.2604.
- [15] R. Alfarizi, A. M. H. Pardede, H. Khair, and S. Kaputama, "Journal of Artificial Intelligence and Engineering Applications Random Forest-Based DDOS Detection from Cpanel Logs with Real-Time Notification Integration," vol. 5, no. 1, pp. 2808–4519, 2025, [Online]. Available: <https://ioinformatic.org/>
- [16] A. Purnomo, A. Kurniasih, A. Nuarminah, and S. Hartati, "Peran Artificial Intelligence dalam Deteksi Dini Ancaman Keamanan Jaringan," *Jurnal Minfo Polgan*, vol. 13, no. 2, pp. 2044–2048, 2024, doi: 10.33395/jmp.v13i2.14356.
- [17] A. W. Ningrum, M. P. Aji, T. Informatika, F. Teknik, and U. M. Purwokerto, "Deteksi dan Klasifikasi Ancaman pada Log Serangan Siber Menggunakan Algoritma K- Nearest Neighbor (KNN) dan Random Forest (RF) Threat Detection and Classification in Cyber Attack Logs Using K-Nearest Neighbor (KNN) and Random Forest (RF) Algorithm," vol. x, no. 12, pp. 1–9, 2021.
- [18] P. N. Jember, D. A. Larasati, and U. Jember, "Penerapan Algoritma Random Forest Dan K-Nearest Neighbor Untuk Deteksi Intrusi Pada Dataset Cicans2017," vol. 2, no. 6, pp. 632–640, 2025.
- [19] E. Angulo, L. Lizcano, and J. Marquez, "A Stacking-Based Ensemble Model for Multiclass DDoS Detection Using Shallow and Deep Machine Learning Algorithms," *Applied Sciences (Switzerland)*, vol. 16, no. 2, 2026, doi: 10.3390/app16020578.