

IMPLEMENTASI *FILTERING ALERT* PADA *INTRUSION PREVENTION SYSTEM (IPS)* BERBASIS *SNORT* UNTUK MITIGASI *FLOODING* NOTIFIKASI TELEGRAM

Nishra Anasika¹, Muhammad Syahroni², Rachmawati³

^{1,2,3}Jurusan Teknik Elektro Politeknik Negeri Lhokseumawe

Email: nishraanasika@gmail.com¹, msyahroni@pnl.ac.id², rachma@pnl.ac.id³

Corresponding Author : Muhammad Syahroni

Jurusan Teknik Elektro Politeknik Negeri Lhokseumawe

Email: msyahroni@pnl.ac.id

Abstrak – Meningkatnya intensitas serangan *cyber* menyebabkan sistem deteksi intrusi berbasis Snort menghasilkan *alert* dalam jumlah besar yang memicu *notification flooding* pada Telegram, sehingga menurunkan efektivitas administrator dalam memantau insiden keamanan. Penelitian ini bertujuan mengimplementasikan mekanisme *filtering alert* pada *Intrusion Prevention System (IPS)* berbasis *Snort* untuk mengurangi *notification flooding* tanpa menurunkan kemampuan deteksi maupun pencegahan serangan. Mekanisme yang diterapkan menggunakan *detection_filter* pada *Snort*, skrip *Python* dengan *cooldown* 60 detik dan notifikasi *summary* 30 detik, serta *auto-blocking* berbasis *nftables*. Penelitian menggunakan metode eksperimental dengan pendekatan kuantitatif melalui perbandingan skenario IDS tanpa *filtering alert* dan IPS dengan *filtering alert* dua lapis serta *auto-blocking*. Pengujian dilakukan sebanyak 30 kali pada masing-masing serangan *ICMP Flood*, *TCP SYN Flood*, dan *SSH Brute Force*. Hasil penelitian menunjukkan bahwa mekanisme *filtering alert* dua lapis mampu mereduksi notifikasi Telegram rata-rata sebesar 93,00%, mempertahankan tingkat deteksi sebesar 100%, mencapai keberhasilan *auto-blocking* sebesar 100%, serta menghasilkan rata-rata waktu respons 7,33 detik. Dengan demikian, integrasi *detection_filter*, skrip *Python*, dan *nftables* efektif meningkatkan efisiensi pengelolaan notifikasi sekaligus mempertahankan kemampuan deteksi dan pencegahan serangan.

Kata-kata kunci: *Intrusion Prevention System (IPS)*, *Snort*, *filtering alert*, *detection_filter*, *Telegram Bot*, *nftables*.

Abstract – The increasing intensity of cyber attacks causes Snort-based intrusion detection systems to generate a large number of alerts that trigger notification flooding on Telegram, thereby reducing the effectiveness of administrators in monitoring security incidents. This research aims to implement an alert filtering mechanism on a Snort-based Intrusion Prevention System (IPS) to reduce notification flooding without decreasing detection or attack prevention capability. The mechanism applied uses Snort's *detection_filter*, a Python script with a 60-second cooldown and a 30-second summary notification, as well as *nftables*-based auto-blocking. The research uses an experimental method with a quantitative approach, comparing an IDS scenario without alert filtering to an IPS with two-layer alert filtering and auto-blocking. Testing was carried out 30 times for each of the *ICMP Flood*, *TCP SYN Flood*, and *SSH Brute Force* attacks. The results show that the two-layer alert filtering mechanism reduces Telegram notifications by an average of 93.00%, maintains a 100% detection rate, achieves 100% auto-blocking success, and produces an average response time of 7.33 seconds. Thus, the integration of *detection_filter*, the Python script, and *nftables* effectively improves notification management efficiency while maintaining attack detection and prevention capability.

Keywords: *Intrusion Prevention System (IPS)*, *Snort*, *filtering alert*, *detection_filter*, *Telegram Bot*, *nftables*.

I. PENDAHULUAN

Perkembangan teknologi informasi telah mendorong transformasi digital di berbagai sektor, menjadikan jaringan komputer sebagai infrastruktur kritis pertukaran data [1]. Di Indonesia, penetrasi internet mencapai 79,50% pada 2024, sementara BSSN mencatat lebih dari 3,64 miliar trafik anomali hingga Juli 2025 [2]. Kondisi tersebut menuntut mekanisme keamanan yang mampu mendeteksi dan mencegah berbagai bentuk

serangan, seperti DDoS, SYN Flood, port scanning, dan brute force, secara cepat dan tepat sasaran.

Salah satu pendekatan yang banyak digunakan adalah Intrusion Detection System (IDS) dan Intrusion Prevention System (IPS). Snort merupakan perangkat lunak open-source yang mampu menganalisis trafik jaringan berdasarkan rules. Berbagai penelitian telah mengembangkan Snort untuk mendukung deteksi, pencegahan, dan notifikasi serangan. Barends dkk.

(2021) mengombinasikan Snort, iptables, dan honeypot sebagai IPS [3]. Purnama dkk. (2023) mengintegrasikan Snort dengan WhatsApp dan Telegram untuk notifikasi [4]. Dawamsyach dkk. (2023) menggunakan Fail2ban untuk menangani DDoS dan *brute force* dengan notifikasi Telegram [5]. Safitri (2024) memanfaatkan Snort dan bot Telegram untuk mendeteksi SYN Flood dan Ping of Death [6], sedangkan Abdullah dkk. (2024) mengombinasikan Snort, Fail2ban, honeypot, dan Telegram untuk menangani brute force [7]. Sheeraz dkk. (2024) mengembangkan Snort sebagai IPS inline dengan fokus pada pemrosesan trafik line rate [8].

Penelitian-penelitian tersebut belum berfokus pada pengendalian jumlah alert yang diteruskan ke administrator, padahal alert berulang dalam waktu singkat dapat menyebabkan *notification flooding* dan *alert fatigue* yang menyulitkan prioritasasi informasi. Diperlukan mekanisme yang mengendalikan volume alert tanpa mengurangi kelengkapan informasi maupun kecepatan respons sistem.

Berdasarkan permasalahan tersebut, penelitian ini merancang dan mengimplementasikan mekanisme filtering alert dua lapis pada IPS berbasis Snort untuk memitigasi *notification flooding* pada notifikasi Telegram. Lapisan pertama menggunakan *detection_filter* pada Snort, sedangkan lapisan kedua menggunakan skrip Python dengan mekanisme *cooldown* dan notifikasi *summary*. Sistem juga menerapkan pemblokiran otomatis (*automated response*) menggunakan *nftables* terhadap IP sumber serangan. Pengujian dilakukan terhadap ICMP Flood, TCP SYN Flood, dan SSH Brute Force dengan parameter jumlah alert, jumlah notifikasi, persentase reduksi, waktu respons, serta keberhasilan pemblokiran. Dengan rancangan tersebut, penelitian ini diharapkan menghasilkan sistem yang mampu mendeteksi, menyaring, memberi notifikasi secara terukur, dan mencegah serangan secara otomatis sekaligus mengurangi beban pemantauan administrator.

II. TINJAUAN PUSTAKA

A. Keamanan Jaringan Komputer

Keamanan jaringan komputer merupakan upaya melindungi perangkat, perangkat lunak, dan data dari kerusakan, perubahan, serta kebocoran agar sistem tetap andal. Tujuannya menjaga kerahasiaan (*confidentiality*), integritas (*integrity*), dan ketersediaan (*availability*) data dalam jaringan [9].

B. Jenis Serangan Jaringan

Serangan jaringan komputer secara umum dikelompokkan menjadi serangan pasif dan serangan aktif. Berikut tiga jenis serangan aktif, yaitu ICMP Flood, TCP SYN Flood, dan SSH Brute Force.

1. ICMP Flood

ICMP Flood merupakan serangan *Denial of Service* (DoS) berupa pengiriman paket ICMP *Echo Request*

secara terus-menerus ke *host* target sehingga menghabiskan bandwidth dan sumber daya pemrosesan [10].

2. TCP SYN Flood

TCP SYN Flood memanfaatkan *mekanisme three-way handshake* TCP dengan mengirimkan paket SYN tanpa menyelesaikan ACK, sehingga menciptakan banyak koneksi *half-open* yang menghabiskan sumber daya server dan mengganggu layanan kepada pengguna sah [11].

3. SSH Brute Force

SSH *Brute Force* merupakan serangan yang bertujuan memperoleh kredensial akses SSH dengan mencoba berbagai kombinasi username dan password secara sistematis dan berulang dari IP yang sama [12].

C. Intrusion Detection System (IDS)

IDS adalah perangkat yang memantau trafik jaringan untuk mendeteksi serangan atau pelanggaran kebijakan keamanan dan menghasilkan *alert* secara pasif tanpa melakukan pencegahan langsung, menggunakan *signature-based* atau *anomaly-based detection*, serta dapat berupa *Network-based* (NIDS) atau *Host-based* (HIDS) [1][13].

D. Intrusion Prevention System (IPS)

Intrusion Prevention System (IPS) merupakan pengembangan dari IDS yang tidak hanya mendeteksi, tetapi juga melakukan tindakan pencegahan terhadap trafik yang teridentifikasi sebagai ancaman. Pada implementasi *inline*, IPS memeriksa paket sebelum diteruskan ke tujuan dan dapat melakukan *drop* terhadap paket berbahaya secara *real-time* [8].

E. Snort

Snort adalah perangkat lunak IDS/IPS *open-source* berbasis *signature* yang menginspeksi paket data secara *real-time* dan mencocokkannya terhadap *rules*, terdiri atas *action*, protokol, alamat/port sumber-tujuan, serta *rule options* seperti *msg*, *sid*, *rev*, *classtype*, dan *priority* [7].

F. Fitur *detection_filter* pada Snort

Detection_filter adalah *rule option* bawaan Snort yang menahan pembentukan *alert* hingga jumlah paket dari IP sumber (*track by_src*) atau tujuan (*track by_dst*) yang sama melampaui ambang batas (*count*) dalam rentang waktu tertentu (*seconds*); setiap paket tetap dievaluasi, namun *alert* baru terbentuk setelah ambang terpenuhi [14].

G. Firewall dan *nftables*

nftables merupakan *firewall* Linux berbasis *Netfilter*, penerus *iptables*, yang mengelola aturan IPv4, IPv6, ARP, dan *bridge* dalam satu kerangka kerja [15], serta mendukung struktur data *set* dan fitur *timeout* sehingga sesuai untuk pengelolaan *blacklist* dan pemblokiran otomatis [16].

H. Telegram Bot API sebagai Media Notifikasi

Telegram menyediakan Bot API yang memungkinkan program mengirim dan menerima pesan otomatis melalui HTTPS [17]; integrasi Snort dengan Telegram dilakukan melalui skrip yang membaca log Snort, mencocokkan *Signature ID*, lalu mengirim notifikasi menggunakan *token*, chat ID, dan metode *sendMessage* [13].

I. Python sebagai Bahasa Pemrograman

Python adalah bahasa pemrograman tingkat tinggi dengan *sintaks* sederhana dan pustaka yang luas, sesuai untuk otomasi, pengolahan *log*, dan integrasi antarkomponen keamanan jaringan [18].

J. Persamaan Perhitungan Reduksi *Alert* dan Notifikasi

Efektivitas *filtering alert* dua lapis diukur dari persentase reduksi alert Snort dan notifikasi Telegram antara Skenario 1 dan Skenario 2 menggunakan Persamaan (1) dan (2).

$$\text{Reduksi Alert (\%)} = \frac{\text{Alert}_{\text{sebelum}} - \text{Alert}_{\text{sesudah}}}{\text{Alert}_{\text{sebelum}}} 100\% \quad (1)$$

$$\text{Reduksi Notifikasi (\%)} = \frac{\text{Notif}_{\text{sebelum}} - \text{Notif}_{\text{sesudah}}}{\text{Notif}_{\text{sebelum}}} 100\% \quad (2)$$

Keterangan:

- *Alert* (sebelum) dan *Alert* (sesudah) merupakan jumlah *alert* yang dihasilkan Snort sebelum dan sesudah penerapan *filtering*.
- *Notif* (sebelum) dan *Notif* (sesudah) merupakan jumlah notifikasi Telegram sebelum dan sesudah penerapan *filtering*.

K. Persamaan Perhitungan Waktu Respons Sistem

Waktu respons dihitung sejak paket serangan pertama diterima hingga notifikasi pertama diterima administrator, menggunakan Persamaan (3):

$$\text{Waktu Respons} = \text{Waktu Deteksi} + \text{Waktu Notifikasi} \quad (3)$$

Keterangan:

- Waktu Deteksi adalah durasi hingga *alert* pertama terbentuk
- Waktu Notifikasi adalah durasi dari terbentuknya *alert* pertama hingga notifikasi pertama diterima di Telegram.

III. METODOLOGI

Penelitian menggunakan metode eksperimen kuantitatif yang membandingkan Skenario 1 (tanpa *filtering alert*) dan Skenario 2 (*filtering alert* dua lapis dengan *auto-blocking*) pada lingkungan virtual VMware Workstation yang terkontrol dan terisolasi.

A. Alat dan Bahan Penelitian

Penelitian menggunakan satu unit laptop sebagai host virtualisasi yang menjalankan dua mesin virtual. VM1 sebagai *attacker*, sedangkan VM2 sebagai target sekaligus *host IPS*, dengan spesifikasi pada Tabel 1 dan Tabel 2.

TABEL I
Spesifikasi Perangkat Keras (*Hardware*)

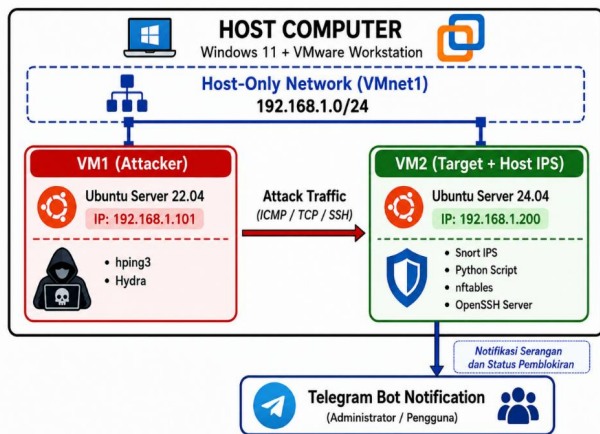
Komponen	Spesifikasi
Processor	AMD Ryzen 5
RAM	8 GB
Penyimpanan	256 GB SSD
Sistem Operasi	Windows 11
Jaringan	<i>Host-Only Network / Internal Network</i>

TABEL II
Spesifikasi Perangkat Lunak (*Software*)

Perangkat Lunak	Detail	Deskripsi
Sistem Operasi <i>Host</i>	Windows 11	Sistem operasi utama pada laptop
Virtualisasi	VMware Workstation	Membuat dan mengelola dua unit VM
OS VM1 (<i>Attacker</i>)	Ubuntu Server 22.04	Sistem operasi mesin penyerang
OS VM2 (Target + IPS)	Ubuntu Server 24.04	Sistem operasi mesin target sekaligus <i>host IPS</i>
IDS/IPS Engine	Snort 2.9.x	Mesin deteksi dan pencegahan berbasis <i>signature</i>
Tools Simulasi Serangan	<i>Hping3, Hydra</i>	Pembangkit trafik serangan pengujian
Layanan Target	OpenSSH Server	Layanan yang disasar pada skenario SSH <i>Brute Force</i>
Firewall	<i>nftables</i>	Mekanisme pemblokiran IP otomatis (fungsi IPS)
Notifikasi	Telegram Bot API	Media pengiriman notifikasi real-time
Bahasa Script	<i>Python3</i>	Pembacaan <i>log</i> Snort, logika <i>filtering</i> , eksekusi blokir

B. Topologi Jaringan Pengujian

Topologi pengujian menggunakan dua mesin virtual yang terhubung melalui *Host-Only Network* pada subnet 192.168.1.0/24. VM1 menggunakan alamat IP 192.168.1.101 sebagai mesin penyerang, sedangkan VM2 menggunakan alamat IP 192.168.1.200 sebagai target sekaligus *host IPS*. Konfigurasi tanpa NAT maupun *bridging* menjaga trafik pengujian tetap berada pada jaringan virtual terisolasi. Telegram Bot API digunakan sebagai jalur pengiriman notifikasi kepada administrator.



Gbr. 1 Rancangan Topologi Jaringan Pengujian

C. Parameter dan Instrumen Pengujian

Data dikumpulkan melalui pengamatan langsung terhadap keluaran *hping3* dan *Hydra*, log Snort, riwayat pesan Telegram, serta status *blacklist nftables*. Parameter yang diamati meliputi jumlah paket atau percobaan, jumlah *alert*, jumlah notifikasi, waktu deteksi, waktu notifikasi, dan keberhasilan pemblokiran IP.

TABEL III
Parameter dan Instrumen Pengujian

Parameter	Instrumen / Cara Pengukuran	Sumber Data
Jumlah paket/percobaan	Statistik <i>hping3</i> dan <i>Hydra</i>	Terminal VM1
Jumlah <i>alert</i>	grep pola pesan /var/log/snort/alert wc -l	Log Snort VM2
Jumlah notifikasi	Hitung pesan pada riwayat Bot	Telegram
Waktu deteksi & notifikasi	Selisih timestamp log Snort dan Telegram	Log + Telegram
Status pemblokiran	nft list set ... dan uji ping	VM2 + VM1

D. Perancangan Filtering Alert Dua Lapis

Mekanisme *filtering* terdiri atas dua lapisan. Lapisan pertama menggunakan *detection_filter* pada aturan Snort untuk menunda pembentukan *alert* sampai jumlah paket atau percobaan dari sumber IP yang sama melewati *threshold* dalam interval 30 detik. Lapisan kedua menggunakan skrip Python *snort_universal.py* yang membaca log Snort, mengidentifikasi *Signature ID*, mengatur *cooldown* 60 detik, mengirimkan satu notifikasi peringatan dan satu notifikasi rangkuman, serta menjalankan pemblokiran IP melalui *nftables*.

Konfigurasi Dasar dan Aturan Snort

Pada VM2, Snort 2.9.x dikonfigurasi dengan HOME_NET 192.168.1.0/24, EXTERNAL_NET !\$HOME_NET, RULE_PATH /etc/snort/rules, dan local.rules sebagai aturan

deteksi kustom. Dua varian aturan digunakan bergantian: aturan tanpa *detection_filter* sebagai baseline Skenario 1 (SPAM) dan aturan dengan *detection_filter* sebagai *filtering* lapisan pertama pada Skenario 2 (FILTER).

Aturan Skenario 1 (tanpa *detection_filter*)

```
# ICMP Flood
alert icmp any any -> $HOME_NET any (msg:"ICMP Flood Detected"; itype:8; priority:3; sid:1000001; rev:1;)

# SYN Flood
alert tcp any any -> $HOME_NET 80 (msg:"SYN Flood Detected"; flags:S; priority:2; sid:1000002; rev:1;)

# SSH Brute Force
alert tcp any any -> $HOME_NET 22 (msg:"SSH Brute Force Detected"; flow:to_server,established; content:"SSH"; priority:1; sid:1000003; rev:1;)
```

TABEL IV
Parameter *detection_filter*

Serangan	Count	Seconds	Justifikasi
ICMP Flood	5 paket	30	Mengurangi false positive dari trafik ping normal
TCP SYN Flood	10 paket	30	Membedakan SYN serangan dari koneksi normal
SSH Brute Force	2 percobaan	30	Indikasi login berulang dari IP yang sama

Aturan Skenario 2 (dengan *detection_filter*)

```
# ICMP Flood
alert icmp any any -> $HOME_NET any (msg:"ICMP Flood Detected"; itype:8; priority:3; detection_filter:track by_src, count 5, seconds 30; sid:1000001; rev:2;)

# SYN Flood
alert tcp any any -> $HOME_NET 80 (msg:"SYN Flood Detected"; flags:S; priority:2; detection_filter:track by_src, count 10, seconds 30; sid:1000002; rev:2;)

# SSH Brute Force
alert tcp any any -> $HOME_NET 22 (msg:"SSH Brute Force Detected"; flow:to_server,established; content:"SSH"; priority:1; detection_filter:track by_src, count 2, seconds 30; sid:1000003; rev:2;)
```

Pada lapisan pertama, *alert* baru terbentuk setelah *threshold detection_filter* terpenuhi, lalu informasi tersebut diteruskan ke skrip *Python* pada lapisan kedua.

E. Pemblokiran Otomatis dan Notifikasi

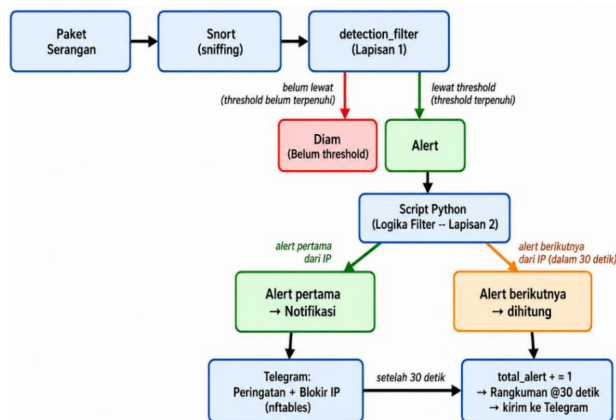
1. Pemblokiran Otomatis dengan *nftables*

Pemblokiran menggunakan tabel filter IPv4, chain INPUT, *set blacklist* dengan fitur *timeout*, dan aturan *drop*. Ketika *alert* pertama terdeteksi pada mode FILTER, skrip *Python* menambahkan IP sumber serangan ke *blacklist* dengan masa berlaku 60 detik.

```
sudo nft add table ip filter
sudo nft add chain ip filter INPUT { type filter
hook input priority 0; policy accept; }
sudo nft add set ip filter blacklist { type
ipv4 addr; flags timeout; }
sudo nft add rule ip filter INPUT ip saddr
@blacklist drop
nft add element ip filter blacklist {<IP> timeout
60s}
```

2. Notifikasi Telegram dan Filtering Lapisan Kedua

Skrip *Python snort_universal.py* membaca `/var/log/snort/alert`, mengenali SID 1000001–1000003, meneruskan informasi kejadian melalui Telegram Bot API, dan menjalankan pemblokiran *nftables*. Mode SPAM digunakan sebagai *baseline*, sedangkan mode FILTER mengirim peringatan pertama per IP dan rangkuman setelah periode pengamatan. *Cooldown* 60 detik dan *summary* 30 detik diterapkan pada mode FILTER.



Gbr. 2 Diagram Alur Mekanisme *filtering alert* Dua Lapis

Pada lapisan kedua, *alert* pertama menjadi pemicu notifikasi peringatan sekaligus *auto-blocking*. *Alert* berikutnya tetap tersimpan pada *log* Snort dan dihimpun sebagai informasi untuk notifikasi rangkuman setelah periode pengamatan 30 detik.

TABEL V
Parameter *Filtering* Dua Lapis

Lapisan	Lokasi	Parameter	Fungsi
1	Snort	<i>count, seconds</i>	Membatasi pembentukan <i>alert</i> hingga <i>threshold</i>
2	<i>Python</i>	<i>cooldown</i> 60 s; <i>summary</i> 30 s	Membatasi notifikasi per IP/periode

F. Skenario dan Prosedur Pengujian

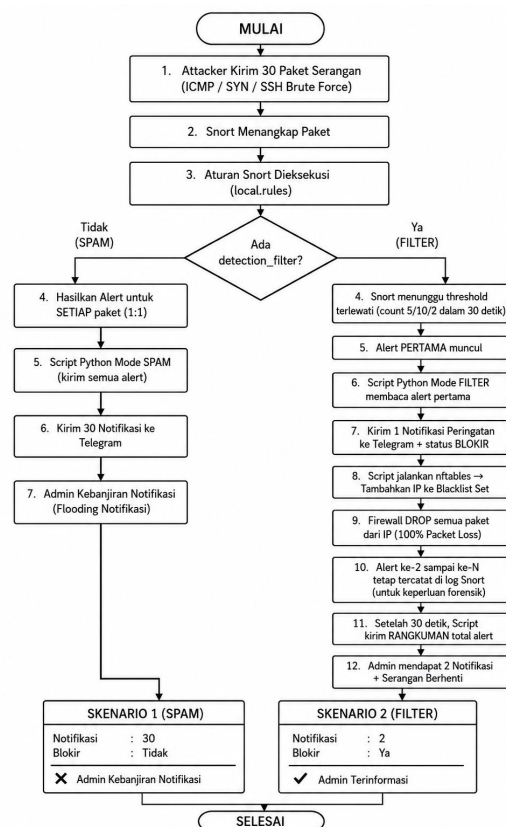
Dua skenario diuji dengan jenis serangan, *tools*, target, dan jumlah percobaan yang sama. Skenario 1 menggunakan IDS/SPAM tanpa *filtering* dan tanpa pemblokiran. Skenario 2 menggunakan IPS/FILTER dengan *detection_filter*, *filtering Python*, notifikasi, serta *auto-blocking*.

TABEL VI
Rancangan Pengujian Penelitian

Skenario	Mode	Jenis Serangan	Jumlah Pengujian	Parameter yang Diamati
1	IDS	ICMP Flood	30 paket	Jumlah <i>alert</i> , jumlah notifikasi Telegram, status pemblokiran IP
		TCP SYN Flood	30 paket	
		SSH Brute Force	30 percobaan login	
2	IPS	ICMP Flood	30 paket	
		TCP SYN Flood	30 paket	
		SSH Brute Force	30 percobaan login	

TABEL VII
Perintah Simulasi Serangan

No	Serangan	Tujuan	Perintah
1	ICMP Flood	30 ICMP Echo Request	<code>sudo hping3 --icmp 192.168.1.200 -c 30</code>
2	TCP SYN Flood	30 SYN ke port 80	<code>sudo hping3 -S 192.168.1.200 -p 80 -c 30</code>
3	SSH Brute Force	30 kombinasi login	<code>hydra -l admin -P 30pass.txt ssh://192.168.1.200 -t 30 -I -V</code>



Gbr. 3 Diagram alur pelaksanaan skenario pengujian.

Gambar 3 menunjukkan bahwa mode IDS (SPAM) meneruskan setiap *alert* sebagai notifikasi tanpa pemblokiran, sedangkan mode IPS (FILTER) memicu notifikasi peringatan dan *auto-blocking* pada alert pertama, dengan alert berikutnya dirangkum setelah 30 detik.

G. Teknik Pengolahan Data

Data mentah diolah menjadi tiga metrik, yaitu persentase reduksi *alert*, persentase reduksi notifikasi, dan total waktu respons. Keberhasilan pemblokiran IP diverifikasi melalui pemeriksaan *blacklist nftables* dan uji konektivitas dari VM1 ke VM2.

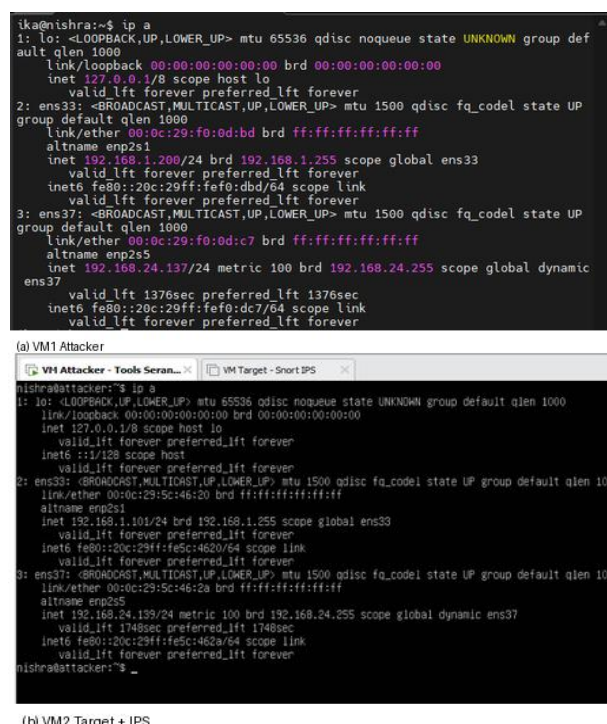
TABEL VIII
Kriteria Keberhasilan Sistem

No.	Kriteria	Target
1	Reduksi jumlah notifikasi Telegram	≥ 90%
2	Seluruh serangan uji terdeteksi	100%
3	IP penyerang berhasil diblokir otomatis	100%
4	Notifikasi informatif	Terpenuhi
5	Total waktu respons (deteksi + notifikasi)	< 10 detik

IV. HASIL DAN PEMBAHASAN

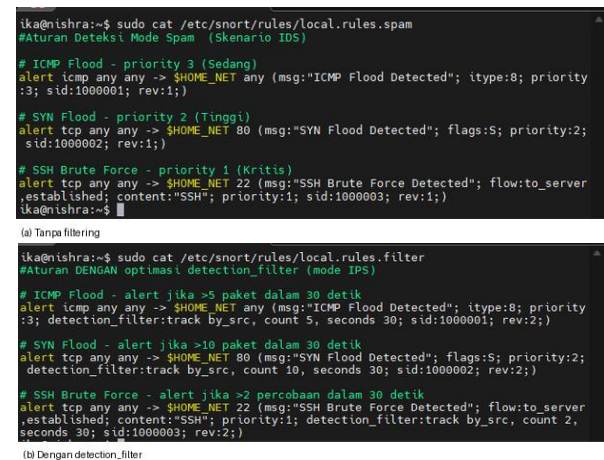
A. Implementasi Sistem

Implementasi sistem dilakukan pada dua mesin virtual dengan Snort sebagai komponen deteksi, Python sebagai pengolah *alert*, Telegram sebagai media notifikasi, dan *nftables* sebagai mekanisme pemblokiran otomatis. Dua varian aturan Snort digunakan untuk membandingkan kondisi tanpa dan dengan *filtering alert*. Implementasi lingkungan pengujian dan aturan Snort ditunjukkan pada Gambar 4 dan Gambar 5.



Gbr. 4 Lingkungan VMware Workstation: (a) VM1 attacker, (b) VM2 target sekaligus *host IPS*

Gambar 4 menunjukkan lingkungan virtual pengujian (VM1 sebagai attacker sedangkan VM2 sebagai target/host IPS); aturan Snort Skenario 1 tanpa *detection_filter*, Skenario 2 dengan *detection_filter* sesuai threshold masing-masing serangan.

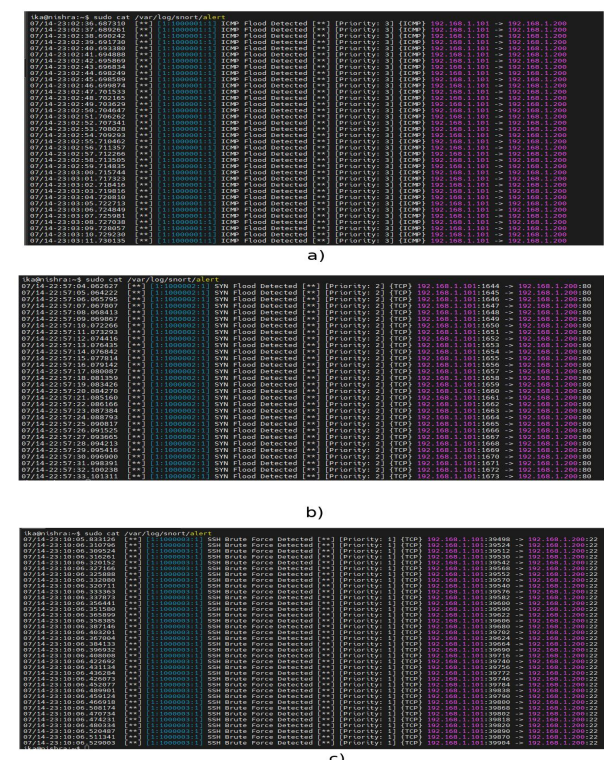


Gbr. 5 Perbandingan aturan Snort tanpa *filtering* dan dengan *detection_filter*

Gambar 5 menunjukkan perbedaan konfigurasi aturan pada kedua skenario, yang menjadi dasar penerapan mekanisme *filtering alert*.

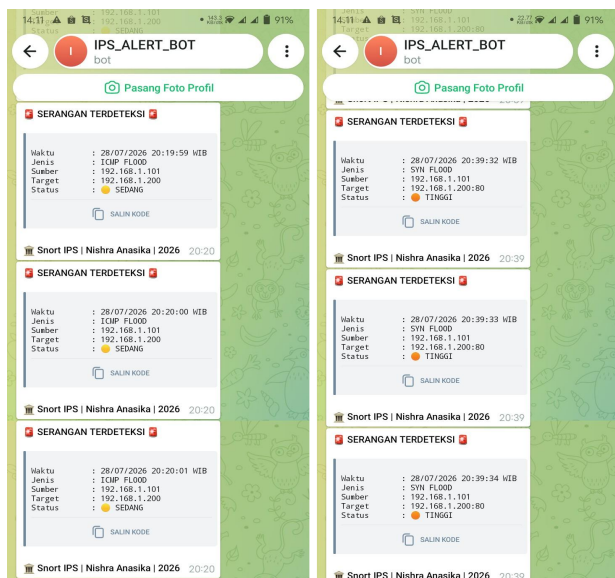
B. Hasil Pengujian Skenario 1: Tanpa *Filtering Alert*

Skenario 1 digunakan sebagai *baseline* untuk mengevaluasi sistem sebelum penerapan *filtering alert*. Pada skenario ini, setiap *alert* yang dihasilkan Snort diteruskan ke Telegram tanpa pemblokiran IP.



Gbr. 6 Log *alert* Snort pada Skenario 1: (a) ICMP Flood, (b) TCP SYN Flood, dan (c) SSH Brute Force

Gambar 6 memperlihatkan *alert* yang tercatat selama serangan. Kondisi tersebut menghasilkan notifikasi telegram secara beruntun, seperti yang ditunjukkan pada Gambar 7.



Gbr. 7 Sampel Notifikasi Telegram pada Skenario 1 yang menunjukkan *notification flooding*

Jumlah *alert* dan notifikasi yang dihasilkan pada setiap jenis serangan disajikan pada Tabel 9.

TABEL IX
Hasil Pengujian Skenario 1 tanpa *Filtering Alert*

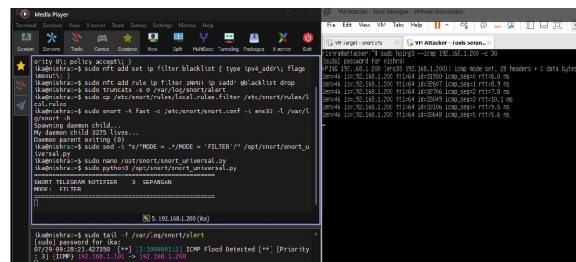
Serangan	Paket/Uji	Alert	Telegram	Blocking
ICMP Flood	30	30	30	Tidak
TCP SYN Flood	30	30	30	Tidak
SSH Brute Force	30	26	26	Tidak

Berdasarkan Tabel IX, ICMP Flood dan TCP SYN Flood menghasilkan masing-masing 30 *alert* dan 30 notifikasi, sedangkan SSH Brute Force menghasilkan 26 *alert* dan 26 notifikasi akibat *rate limiting* pada layanan SSH. Hasil ini menjadi *baseline* untuk dibandingkan dengan Skenario 2.

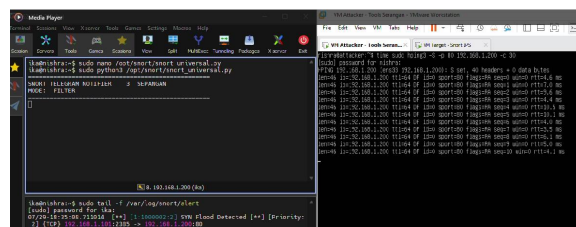
C. Hasil Pengujian Skenario 2: Dengan *Filtering Alert*

Pada Skenario 2, *detection_filter*, skrip Python mode FILTER, dan *nftables* diaktifkan. Jenis serangan, *tools*, dan jumlah pengujian disamakan dengan Skenario 1 untuk menjaga kondisi perbandingan.

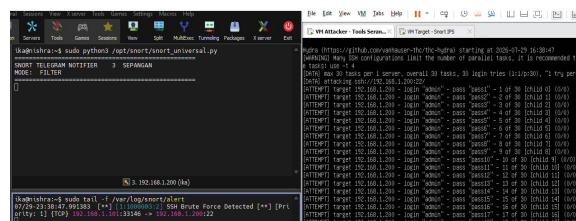
Gambar 8 menunjukkan pencatatan *alert* setelah penerapan *detection_filter*. Hasil pengujian berdasarkan masing-masing *threshold* dirangkum pada Tabel 10.



(a)



(b)



(c)

Gbr. 8 Log alert Snort setelah *filtering*: (a) ICMP Flood, (b) TCP SYN Flood, dan (c) SSH Brute Force

TABEL X
Hasil Pengujian Skenario 2 dengan *Filtering Alert*

Serangan	Threshold	Uji	Alert	Telegram	Blocking
ICMP Flood	5/30 s	30	25	2	Ya
TCP SYN Flood	10/30 s	30	20	2	Ya
SSH Brute Force	2/30 s	30	25	2	Ya

Berdasarkan Tabel 10, penerapan *filtering* menghasilkan 25 *alert* pada ICMP Flood, 20 pada TCP SYN Flood, dan 25 pada SSH Brute Force. Jumlah notifikasi pada ketiga serangan dibatasi menjadi dua pesan.



Gbr. 9 Notifikasi Telegram setelah *filtering*

Gambar 9 menunjukkan bahwa notifikasi dibatasi menjadi pesan peringatan dan rangkuman, sehingga informasi utama serangan tetap disampaikan tanpa meneruskan seluruh *alert* sebagai pesan terpisah.

D. Analisis Reduksi *Alert* dan Notifikasi

Perbandingan Skenario 1 dan Skenario 2 digunakan untuk mengetahui pengaruh *filtering alert* terhadap jumlah *alert* Snort dan notifikasi Telegram. Hasil perhitungan berdasarkan Persamaan (1) dan (2) disajikan pada Tabel 11.

TABEL XI
Hasil *Filtering Alert* dan Notifikasi

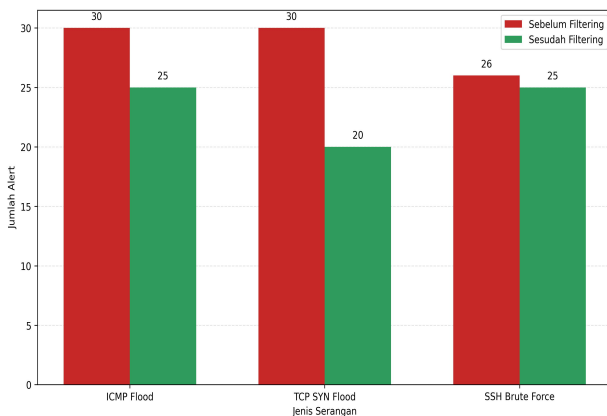
Jenis Serangan	Alert S1 / S2	Reduksi Alert	Notifikasi S1 / S2	Reduksi Notifikasi
ICMP Flood	30 / 25	16,67%	30 / 2	93,33%
TCP SYN Flood	30 / 20	33,33%	30 / 2	93,33%
SSH Brute Force	26 / 25	3,85%	26 / 2	92,31%
Rata - rata	28,67 / 23,33	17,95%	28,67 / 2	93,00%

Keterangan:

S1 = Skenario 1 tanpa *filtering*

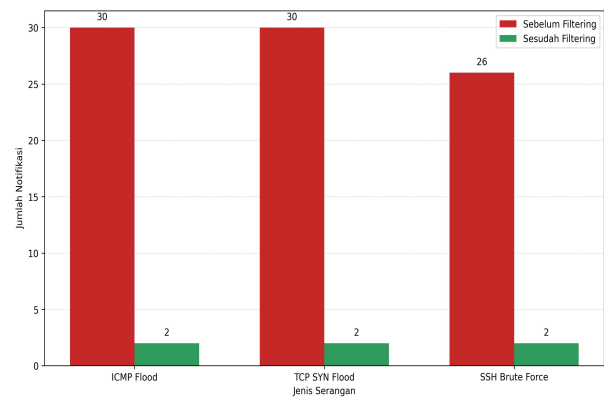
S2 = Skenario 2 dengan *filtering*

Berdasarkan Tabel 11, reduksi *alert* rata-rata mencapai 17,95%, sedangkan reduksi notifikasi Telegram mencapai 93,00% menunjukkan bahwa mekanisme dua lapis memberi pengaruh utama pada pengendalian pesan ke administrator.



Gbr. 10 Perbandingan Jumlah *alert* Snort sebelum dan sesudah *Filtering*

Gambar 10 menunjukkan penurunan jumlah *alert* setelah *filtering*, dengan reduksi terbesar pada TCP SYN Flood sebesar 33,33% (30 menjadi 20). ICMP Flood berkurang 16,67% (30 menjadi 25), sedangkan SSH Brute Force 3,85% (26 menjadi 25). Hasil ini menunjukkan bahwa *detection_filter* membatasi pembentukan *alert* sesuai *threshold*.



Gbr. 11 Perbandingan Jumlah Notifikasi Telegram sebelum dan sesudah *filtering*

Gambar 11 menunjukkan penurunan notifikasi yang signifikan pada seluruh jenis serangan, menegaskan bahwa *cooldown* dan *summary* efektif mengendalikan notifikasi berulang.

E. Waktu Respons Sistem

Waktu respons diukur sejak *alert* pertama dihasilkan Snort hingga notifikasi pertama diterima di Telegram, sesuai Persamaan (3), dan dirangkum pada Tabel 12.

TABEL XII
Waktu Respons Sistem per Jenis Serangan

Jenis Serangan	Threshold (count)	Waktu Deteksi (detik)	Waktu Notifikasi (detik)	Total Respons (detik)
ICMP Flood	5	6	1	7
TCP SYN Flood	10	11	1	12
SSH Brute Force	2	2	1	3
Rata-rata		6,33	1	7,33

Waktu notifikasi relatif konstan sekitar 1 detik pada seluruh jenis serangan, menunjukkan proses pengiriman notifikasi melalui *Telegram Bot API* berlangsung stabil. Perbedaan waktu respons terutama berasal dari waktu deteksi yang dipengaruhi *threshold*. SSH Brute Force (count 2) memiliki waktu deteksi tercepat sebesar 2 detik, diikuti ICMP Flood (count 5) sebesar 6 detik dan TCP SYN Flood (count 10) sebesar 11 detik. Rata-rata total respons sebesar 7,33 detik masih memenuhi target <10 detik.

F. Verifikasi *Auto-Blocking*

Verifikasi *auto-blocking* dilakukan dengan memeriksa *blacklist set* pada *nftables* dan menguji konektivitas dari VM1 ke VM2 setelah pemblokiran.

```
ika@nishra:~$ sudo cat /var/log/snort/alert | grep ICMP | wc -l
25
ika@nishra:~$ sudo nft list set ip filter blacklist
table ip filter {
    set blacklist {
        type ipv4_addr
        flags timeout
        elements = { 192.168.1.101 }
    }
}
```

(a)

```
nishra@attacker:~$ ping 192.168.1.200
PING 192.168.1.200 (192.168.1.200) 56(84) bytes of data.
--- 192.168.1.200 ping statistics ---
19 packets transmitted, 0 received, 100% packet loss, time 18418ms
```

(b)

Gbr. 13 Verifikasi *auto-blocking* melalui: (a) *blacklist nftables*, (b) Hasil Uji Konektivitas

Gambar 13 menunjukkan IP sumber 192.168.1.101 terdaftar pada *blacklist* dan konektivitas ke target mengalami 100% *packet loss*. Hasil pada ketiga jenis serangan ditunjukkan pada Tabel 13.

TABEL XIII
Verifikasi Pemblokiran IP dengan *nftables*

Jenis Serangan	IP Sumber	Status pada <i>Blacklist</i>	Hasil Uji Konektivitas
ICMP Flood	192.168.1.101	Terdaftar	100% <i>packet loss</i>
TCP SYN Flood	192.168.1.101	Terdaftar	100% <i>packet loss</i>
SSH Brute Force	192.168.1.101	Terdaftar	100% <i>packet loss</i>

Berdasarkan Tabel 13, IP sumber 192.168.1.101 berhasil diblokir pada seluruh jenis serangan dengan hasil uji konektivitas 100% *packet loss*.

G. Evaluasi Kriteria Keberhasilan

Berdasarkan hasil pengujian, pencapaian sistem terhadap kriteria yang telah ditetapkan, dirangkum pada Tabel 14.

TABEL XIV
Evaluasi Kriteria Keberhasilan Sistem

Kriteria Keberhasilan	Target	Hasil	Status
Reduksi notifikasi Telegram	$\geq 90\%$	93,00%	Berhasil
Seluruh serangan terdeteksi	100% terdeteksi	100% terdeteksi	Berhasil
IP penyerang diblokir otomatis	100% diblokir	100% diblokir	Berhasil
Notifikasi tetap informatif	Terpenuhi	Ya	Berhasil
Total waktu respons sistem	< 10 detik	Rata-rata 7,33 detik	Berhasil

Secara keseluruhan, *filtering alert* dua lapis berhasil mengurangi *notification flooding* tanpa menurunkan

kemampuan deteksi maupun pemblokiran otomatis pada ketiga jenis serangan, dengan notifikasi turun rata-rata 93,00%, waktu respons rata-rata 7,33 detik, dan keberhasilan pemblokiran 100%.

V. KESIMPULAN

Berdasarkan hasil penelitian dan pengujian, dapat disimpulkan bahwa Skenario IDS tanpa *filtering alert* menghasilkan 30 *alert* dan 30 notifikasi pada ICMP Flood dan TCP SYN Flood, serta 26 *alert* dan 26 notifikasi pada SSH Brute Force. Sistem berhasil diimplementasikan sebagai IPS menggunakan *detection filter*, *cooldown* 60 detik, *summary notification* 30 detik, dan *auto-blocking* berbasis *nftables*. Penerapan *filtering alert* dua tahap mampu mengurangi notifikasi Telegram sebesar 93,00% dan *alert* Snort sebesar 17,95%, dengan tingkat deteksi dan keberhasilan *auto-blocking* mencapai 100% serta rata-rata waktu respons 7,33 detik.

Secara ilmiah, penelitian ini memberikan kontribusi penting bagi komunitas keamanan siber berupa model empiris mitigasi *alert fatigue* melalui mekanisme *filtering* dua tahap yang terbukti tidak menurunkan akurasi deteksi. Metode ini dapat menjadi referensi akademis baru dalam pengembangan arsitektur IDS/IPS yang lebih efisien dan adaptif. Dari sisi ekonomi, implementasi sistem berbasis open-source (Snort dan *nftables*) ini menawarkan implikasi biaya yang sangat signifikan. Pengurangan notifikasi hingga 93% secara langsung memangkas biaya operasional (OpEx) *overhead* jaringan, bandwidth API, dan waktu kerja analisis keamanan.

REFERENSI

- [1] Ardiyansyah, F., Setiawan, K., & Sutisna, N. Implementation of IDS on Computer Networks Using Snort Based on Telegram Chatbot Implementasi IDS pada Jaringan Komputer Menggunakan Snort Berbasis Chatbot Telegram. *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, 4(4), 1614-1623.
- [2] Dewantara, R., Nufus, G. K. ., Pranata, E. J. ., & Djati, F. N. . (2026). Analysis of National Security Mitigation Strategy Effectiveness Against 2025 Cyber Warfare Escalation. *Journal of Applied Informatics Science*, 2(1), 13-18. <https://doi.org/10.65897/jais.v2.i1.80>.
- [3] Barends, J. K., Dewanta, F., & Karna, N. B. A. (2022). Perancangan dan Analisis Intrusion Prevention Sistem Berbasis SNORT dan IPTABLES dengan Integrasi Honeypot pada Arsitektur Software Defined Network. *MULTINETICS*, 7(2), 163–176. <https://doi.org/10.32722/multinetics.v7i2.4276>.
- [4] Purnama, T., Muhyidin, Y., & Singasatia, D. (2023). Implementasi Intrusion Detection System

- (IDS) Snort sebagai Sistem Keamanan menggunakan Whatsapp dan Telegram sebagai Media Notifikasi. *Jurnal Teknologi Informasi dan Komunikasi*, 14(2), 358-369. <https://doi.org/10.51903/jtikp.v14i2.726>.
- [5] Dawamsyach, F., Ruslianto, I., & Ristian, U. (2023). Implementation of IPS (Intrusion Prevention System) Fail2ban on Server for DDoS and Brute Force Attacks. *CESS (Journal of Computer Engineering, System and Science)*, 8(1), 149-161. <https://doi.org/10.24114/cess.v8i1.40259>.
- [6] Safitri, Z. A. (2024). Intrusion Detection System (IDS) pada Snort Dengan Bot Telegram sebagai Sistem Notifikasi Terhadap Serangan Syn Flood dan Ping of Death. *SATIN: Sains dan Teknologi Informasi*, 10(1), 157-167.
- [7] Abdullah, R. K., Fudhail, M. T., & Mujahidin, S. (2024). Penggunaan Snort dan Fail2Ban sebagai IDS untuk mengatasi brute force attack dengan notifikasi Telegram: Studi kasus pada institusi XYZ. *JUSTIN (Jurnal Sistem dan Teknologi Informasi)*, 12(3), 530-542. <https://doi.org/10.26418/justin.v12i3.79617>.
- [8] Sheeraz, M., Durad, M. H., Tahir, S., Tahir, H., Saeed, S., & Almuhaideb, A. M. (2024). Advancing snort ips to achieve line rate traffic processing for effective network security monitoring. *IEEE Access*, 12, 61848-61859.
- [9] Munawar, Z., & Putri, N. I. (2020). Keamanan Jaringan Komputer Pada Era Big Data. *J-SIKA|Jurnal Sistem Informasi Karya Anak Bangsa*, 2(01), 14-20. <https://ejournal.unibba.ac.id/index.php/j-sika/article/view/275>.
- [10] Riyadhi, Y. (2019). *Deteksi Serangan ICMP Flood di Cloud Computing dengan Metode Rule Based Signature*. Tugas Akhir, Jurusan Sistem Komputer, Fakultas Ilmu Komputer, Universitas Sriwijaya.
- [11] Ma'arif, A. M. S. E., Fattah, F., & Darwis, H. (2025). Analisis Perbandingan Serangan UDP Flooding dan SYN Flooding Menggunakan Metode Support Vector Machine. *LNIER: Literatur Informatika dan Komputer*, 2(2), 147-159.
- [12] Januantoro, A., & Supangat, S. (2025). Deteksi Serangan Jaringan Komputer Berbasis Snort dengan Integrasi Notifikasi Real-Time Melalui Telegram. *Jurnal Mnemonic*, 8(1), 100-105.
- [13] Furqan, N., Suandi, I., & Muhammad, M. (2023). Implementasi Intrusion Detection System (IDS) pada Sistem Keamanan Jaringan Menggunakan Telegram sebagai Media Notifikasi. *Jurnal TEKTRON*, 7(1), 44-50.
- [14] Snort Team. (n.d.). README.filters. Snort Documentation. Diakses 3 Mei 2026, dari <https://www.snort.org/document/readme-filters>.
- [15] Netfilter Project. (n.d.). What is nftables? nftables Wiki. Diakses 3 Agustus 2026, dari <https://wiki.nftables.org/>.
- [16] Purwoko, M., & Hilal, H. (2019). Analisis Penerapan Firewall Nftables Sebagai Sistem Keamanan Server Pada Mesin Virtualisasi. *InComTech : Jurnal Telekomunikasi Dan Komputer*, 9(1), 1-22. <https://doi.org/10.22441/incomtech.v9i1.5676>.
- [17] Fahana, J., Umar, R., & Ridho, F. (2017). Pemanfaatan Telegram sebagai Notifikasi Serangan untuk Keperluan Forensik Jaringan. *Query: Journal of Information Systems*, 1(2), 6-14.
- [18] Python Software Foundation. (n.d.). The Python Tutorial. Python 3 Documentation. Diakses 3 Mei 2026, dari <https://docs.python.org/3/tutorial/>.