

ANALISIS KINERJA SURICATA TERHADAP SERANGAN BRUTE FORCE PADA SERVER SSH

Ivan Jacobus Hutaauruk¹, Tengku Mohd Diansyah²

^{1,2}Program Studi Teknik Informatika, Fakultas Teknik dan Komputer, Universitas Harapan
Medan, Medan, Indonesia

e-mail: ¹ivanhutaauruk6@gmail.com, ²dian.10.22@gmail.com

ABSTRAK

Layanan *Secure Shell* (SSH) rentan terhadap serangan *brute force*, namun literatur masih minim menganalisis mekanisme internal yang memengaruhi *Detection Rate* (DR) di bawah serangan berkecepatan tinggi. Penelitian ini bertujuan menganalisis kinerja Suricata Network *Intrusion Detection System* (NIDS) pada Ubuntu Server 24.04 LTS menggunakan metode eksperimental kuantitatif. Suricata menginspeksi lalu lintas port 22 secara *real-time* via aturan berbasis *signature*. Pengujian diawali evaluasi *False Positive Rate* (FPR), dilanjutkan dua skenario: Skenario I (*baseline*) dan Skenario II (Suricata). Hasil FPR 0,000% (*True Negative* (TN) = 4, *False Positive* (FP) = 0). Pada Skenario I, Suricata tidak diaktifkan sehingga proses deteksi tidak terjadi. DR Skenario II mencatatkan $86,969 \pm 0,801\%$. Capaian DR di bawah 100% dipicu penolakan koneksi sepihak OpenSSH akibat terlampauinya batas *MaxStartups* (10:30:100) memicu paket TCP RST/RST-ACK, memutus sesi sebelum Suricata selesai merekonstruksi paket data. *Throughput* serangan 19,677 Kbps (Skenario I) dan 19,948 Kbps (Skenario II). *Packet loss* 0,000% (Skenario I) dan 0,053% (Skenario II). Kenaikan *Central Processing Unit* (CPU) 3,099% (Skenario I) dan 4,332% (Skenario II), serta kenaikan *Random Access Memory* (RAM) 25,732 MB (Skenario I) dan 18,162 MB (Skenario II), dengan *overhead* RAM *idle* 48,035 MB (2,345% dari total RAM). Suricata terbukti presisi dan efisien, namun memerlukan mitigasi aktif tambahan.

Kata kunci: *brute force*; Suricata; NIDS; keamanan jaringan; Ubuntu Server.

ABSTRACT

Secure Shell (SSH) services are vulnerable to brute force attacks, yet literature remains limited in analyzing internal mechanisms affecting *Detection Rate* (DR) under high-speed attacks. This study analyzes Suricata Network *Intrusion Detection System* (NIDS) performance on Ubuntu Server 24.04 LTS using a quantitative experimental method. Suricata inspects port 22 traffic in *real-time* via *signature-based* rules. Evaluation began with *False Positive Rate* (FPR) testing, followed by two scenarios: Scenario I (*baseline*) and Scenario II (Suricata). Results showed FPR 0.000% (*True Negative* (TN) = 4, *False Positive* (FP) = 0). In Scenario I, Suricata was inactive, so no detection occurred. Scenario II recorded a DR of $86.969 \pm 0.801\%$. DR below 100% was triggered by OpenSSH connection refusals when exceeding *MaxStartups* (10:30:100) sending TCP RST/RST-ACK packets, terminating sessions before Suricata finished reassembling packet data. *Throughput* was 19.677 Kbps (Scenario I) and 19.948 Kbps (Scenario II). *Packet loss* 0.000% (Scenario I) and 0.053% (Scenario II). *Central Processing Unit* (CPU) load was 3.099% (Scenario I) and 4.332% (Scenario II), and *Random Access Memory* (RAM) 25.732 MB (Scenario I) and 18.162 MB (Scenario II), with *overhead* RAM *idle* 48.035 MB (2.345% of total RAM). Suricata proved precise and efficient, but requires active mitigation.

(Scenario I) and 4.332% (Scenario II), while Random Access Memory (RAM) load was 25.732 MB (Scenario I) and 18.162 MB (Scenario II), with idle RAM overhead 48.035 MB (2.345% of total RAM). Suricata proves precise and efficient, yet requires active mitigation.

Keywords: brute force; Suricata; NIDS; network security; Ubuntu Server

1. PENDAHULUAN

Perkembangan teknologi informasi menuntut tersedianya sistem yang tidak hanya cepat dan andal, tetapi juga menjamin tingkat keamanan tinggi dalam setiap proses pertukaran data [1]. Sebagai pusat kendali yang memfasilitasi interaksi pengguna di dalam jaringan, *server* memegang peran krusial yang turut membawa kerentanan untuk dieksploitasi [2], [3]. Saat ini, pengelolaan *server* mayoritas dioperasikan melalui akses jarak jauh guna memberikan efisiensi tanpa memerlukan kehadiran fisik administrator [4]. Meskipun protokol *Secure Shell* (SSH) dirancang sebagai jalur utama yang aman untuk kebutuhan tersebut, layanan ini kerap menjadi target utama eksploitasi siber, khususnya *brute force* [4], [5]. Apabila upaya ini berhasil menembus autentikasi, penyerang akan memperoleh hak akses ilegal yang berimplikasi pada penguasaan *server* secara penuh dan kompromi terhadap informasi sensitif di dalamnya. Oleh karena itu, diperlukan sebuah mekanisme perlindungan proaktif yang sanggup mengidentifikasi anomali jaringan sebelum terjadi kerusakan yang lebih fatal [4], [6].

Salah satu instrument utama dalam mendeteksi ancaman jaringan adalah *Network Intrusion Detection System* (NIDS) [7]. NIDS beroperasi dengan mengawasi lalu lintas jaringan guna menemukan aktivitas yang terindikasi mencurigakan dan membangkitkan peringatan keamanan [7], [8]. Diantara berbagai perangkat lunak NIDS *open-source*, Suricata menonjol sebagai mesin inspeksi keamanan jaringan berkecepatan tinggi yang dikembangkan oleh Open Information Security Foundation (OISF). Suricata dibekali kapabilitas pemrosesan *multi-threading* serta kemampuan analisis *Deep Packet Inspection* (DPI) pada aliran protokol TCP [8], [9]. Berbagai penelitian terdahulu telah membuktikan kapabilitas Suricata dalam menginspeksi jaringan; [10] dan [11] menunjukkan bahwa Suricata dapat mendeteksi ancaman jaringan secara *real-time* dengan waktu respons yang sangat responsif. Sementara itu, [8] dan [12] mengevaluasi efektivitas deteksi berbasis indikator *Detection Rate* (DR) dan *False Positive Rate* (FPR), yang membuktikan keunggulan Suricata dalam melakukan inspeksi paket di lingkungan ter-virtualisasi. Selain itu, [1] dan [13] mengonfirmasi kelayakan penerapan sistem pengamanan berbasis Suricata pada sistem operasi linux ubuntu.

Dalam upaya pengambilalihan *server* secara paksa, *brute force* menjadi vektor serangan yang paling umum menargetkan protokol SSH [4], [5]. *Brute force* bekerja dengan mengeksploitasi protokol SSH melalui percobaan tebakan kombinasi kredensial secara masif dan teromatisasi, seperti pemanfaatan instrumen THC-Hydra [2], [5]. Tanpa adanya sistem pemantauan yang mampu mengenali pola inisiasi koneksi abnormal, gelombang serangan ini dapat membebani antrean koneksi *server* bahkan dapat menembus autentikasi [5].

Meskipun penelitian terdahulu telah banyak mengulas Suricata pada berbagai protokol jaringan, mayoritas literatur berfokus pada deteksi trafik umum, tanpa menganalisis dinamika yang memengaruhi Suricata ketika dibawah serangan *brute force*

SSH berkecepatan tinggi. Meninjau celah penelitian tersebut, penelitian ini dilaksanakan untuk merancang, mengimplementasikan, serta menganalisis kinerja Suricata pada Ubuntu Server 24.04 LTS, sekaligus menguraikan faktor-faktor yang memengaruhi capaian *Detection Rate* Suricata. Evaluasi kinerja difokuskan pada pengukuran tingkat efektivitas deteksi (*Detection Rate* dan *False Positive Rate*), laju transmisi serangan (*throughput*), kualitas layanan jaringan berupa *packet loss*, serta dampak terhadap sumber daya server (utilisasi CPU dan RAM).

2. METODE PENELITIAN

Jenis Penelitian

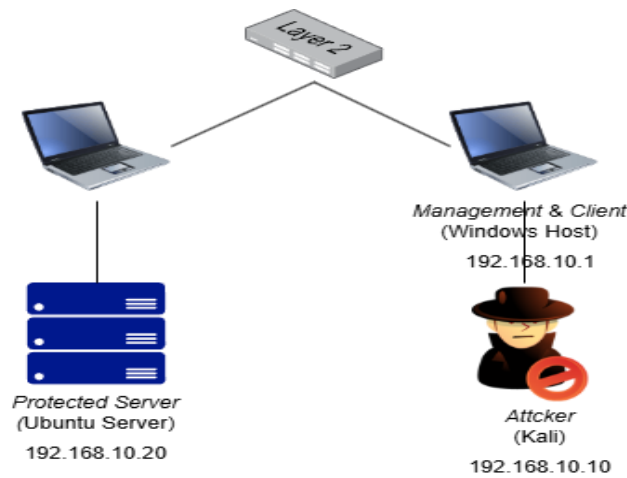
Penelitian ini menggunakan pendekatan kuantitatif dengan metode eksperimen yang dilaksanakan dalam lingkungan laboratorium. Pendekatan ini dipilih secara khusus untuk menguji performa dan efektivitas mesin inspeksi Suricata secara objektif melalui pengukuran metrik terkuantifikasi (*Detection Rate*, *False Positive Rate*, *Throughput* serangan, *Packet Loss*, serta beban komputasi (CPU dan RAM) pada lingkungan sistem operasi Ubuntu Server 24.04 LTS.

Pemilihan metode eksperimental pada laboratorium terisolasi didasarkan pada kebutuhan untuk mengisolasi lingkungan pengujian dari trafik pengganggu (*network noise*) luar, sehingga seluruh data statistik yang terekam murni merefleksikan interaksi antara gelombang serangan *brute force* SSH dan beban pemrosesan *Deep Packet Inspection* (DPI) Suricata. Selain itu, pengujian komparatif diulang sebanyak 5 kali iterasi untuk menjamin validitas data melalui perhitungan nilai rata-rata dan simpangan baku (*mean $\pm$ standard deviation*).

Topologi Jaringan dan Pengalamatan IP Statis

Penelitian ini dilaksanakan pada jaringan lokal terisolasi berbasis *bridged adapter* untuk menjamin bahwa seluruh data eksperimen bebas dari interferensi trafik luar. Topologi jaringan terdiri dari 3 *node* utama dengan pengalamatan IP statis pada subnet 192.168.10.0/24:

1. *Protected Server* (Ubuntu Server 24.04 LTS): Alamat IP 192.168.10.20, menjalankan daemon OpenSSH pada *port* 22 dan Suricata.
2. *Attacker* (Kali Linux 2026.2): Alamat IP 192.168.10.10, bertindak sebagai penyerang yang mengeksekusi skrip otomatisasi THC-Hydra 9.6.
3. *Management & Client* (Windows *Host*): Alamat IP 192.168.10.1, bertindak sebagai pengguna sah yang melakukan akses *login* legal untuk pengujian *False Positive Rate* (FPR) dan *packet loss*.



Gambar 1. Topologi Jaringan Pengujian Terisolasi

Spesifikasi Perangkat dan Perangkat Lunak

Lingkungan pengujian divirtualisasikan menggunakan Oracle VM VirtualBox pada dua unit perangkat keras fisik (*Host*) yang saling terhubung menggunakan *Switch Layer 2*. Mesin virtual *Protected Server* (Ubuntu Server 24.04 LTS) yang menjalankan layanan SSH dan Suricata dibuat dan dijalankan diatas Laptop HP 14s. Sementara itu, mesin virtual *Attacker* (Kali Linux 2026.02) yang mengeksekusi skrip THC-Hydra serta *Client* (Windows *Host*) dibuat dan dijalankan di atas Laptop Infinix XBOOK B14. Pemisahan *host* fisik ini diterapkan untuk mensimulasikan kondisi lingkungan jaringan riil secara terisolasi dan memastikan pengukuran beban komputasi *server* terbebas dari dampak eksekusi serangan di sisi penyerang. Spesifikasi rinci alokasi mesin virtual dan perangkat host ditunjukkan dalam Tabel 1.

Tabel 1. Spesifikasi Perangkat Pengujian

Parameter Virtualisasi	<i>Attacker Node</i>	<i>Protected Server</i> (Ubuntu <i>Server</i>)
Sistem Operasi	Kali Linux 2026.02	Ubuntu <i>Server</i> 24.04 LTS
Alokasi Memori (RAM)	2048 MB (2 GB)	2048 MB (2 GB)
Penyimpanan Virtual (<i>Virtual Disk</i>)	85 GB	85 GB
<i>Network Adapter</i>	<i>Bridged Adapter</i>	<i>Bridged Adapter</i>
Perangkat Keras Fisik (<i>Host</i>)	Laptop Infinix XBOOK B14 (Ryzen 5 7535HS, RAM 16 GB DDR5, ROM 512 GB SSD NVMe)	Laptop HP 14s (Ryzen 7 5825U, RAM 8 GB DDR4, ROM 512 GB SSD NVMe)

Konfigurasi Aturan Suricata

Aturan kustom Suricata dikonfigurasi untuk mengidentifikasi pola inisiasi koneksi SSH berulang sesuai spesifikasi RFC 4253 [14] sebagai berikut:

alert tcp-stream any any -> \$HOME_NET 22 (msg:"Potential SSH Brute Force Activity (High-Frequency SSH Connections)"; flow:established,to_server; content:"SSH-"; depth:4; fast_pattern; detection_filter: track by_src, count 10, seconds 60; classtype:attempted-admin; sid:1000002; rev:9;)

Skenario Pengujian dan Formulasi Parameter Evaluasi

Pengujian dievaluasi dalam ruang lingkup jaringan lokal terisolasi dan diklasifikasikan ke dalam dua tahapan pengujian:

Pengujian False Positive Rate (FPR), dilakukan lebih awal dan dipisahkan secara khusus agar evaluasi akurasi sistem terhadap pengguna sah (*Legitimate User*). Pengujian akan dilakukan menjadi 2 skenario yaitu tanpa serangan dan saat diserang *brute force*.

Pengujian Dua Skenario, diulang sebanyak 5 kali iterasi untuk mengukur parameter efektivitas (*Detection Rate*), *Packet Loss*, *Throughput* serangan, dan beban komputasi (CPU dan RAM).

1. Skenario I (Baseline): *Server* beroperasi tanpa sistem pengamanan
2. Skenario II (Deteksi): Suricata NIDS aktif memantau trafik jaringan.

Formulasi matematis untuk masing-masing parameter evaluasi ditentukan sesuai aturan persamaan matematika berikut:

$$DR = (TP \div (TP + FN)) \times 100\% \quad (1)$$

Dengan DR adalah *Detection Rate* (%), TP adalah *True Positive* (paket serangan terdeteksi), dan FN adalah *False Negative* (serangan yang tidak terdeteksi) [8], [12].

$$FPR = (FP \div (FP + TN)) \times 100\% \quad (2)$$

Dengan FPR adalah *False Positive Rate* (%), FP adalah *False Positive* (koneksi sah terblokir), dan TN adalah *True Negative* (Koneksi sah yang lolos) [12].

$$Throughput = (Total\ Data\ (Byte) \times 8) \div Durasi\ Serangan\ (Detik) \quad (3)$$

Dengan *Throughput* serangan diukur dalam satuan Kilobit per detik (Kbps) dari total volume data lalu lintas serangan [15], [16].

$$Packet\ Loss = ((Paket\ Dikirim - Paket\ Diterima) \div Paket\ Dikirim) \quad (4)$$

Dengan *Packet Loss* dihitung dari rasio paket data yang gagal mencapai tujuan selama periode serangan [15], [16].

$$\Delta\ CPU\ (\%) = Rata - rata\ CPU\ Saat\ Serangan - Rata - rata\ Baseline\ CPU \quad (5)$$

Dengan $\Delta\ CPU\ (\%)$ menyatakan selisih persentase kenaikan rata-rata penggunaan CPU saat pengujian terhadap rata-rata kondisi *baseline*.

$$\Delta\ RAM = Rata - rata\ Ram\ Saat\ Serangan - Rata - rata\ Baseline\ RAM \quad (6)$$

Dengan Δ RAM (MB) menyatakan selisih kenaikan rata-rata alokasi RAM saat pengujian serangan terhadap rata-rata kondisi *baseline*.

Tahapan Pengujian *False Positive Rate* (FPR)

1. Inisiasi Akses Sah: Mengaktifkan koneksi jaringan dari Windows *Host* menuju *Protected Server* dengan memastikan daemon OpenSSH (*port* 22) siap menerima sesi autentikasi.

```
root@ubuntuserver:/home/testuser# systemctl enable --now ssh
```

Gambar 2. Perintah Untuk Otomatis Menjalankan Layanan Ssh Saat *Server Booting*

```
root@ubuntuserver:/home/testuser# ufw allow OpenSSH
```

Gambar 3. Perintah Agar Firewall Mengizinkan Koneksi Ke SSH

```
root@ubuntuserver:/home/testuser# systemctl status ssh
* ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/usr/lib/systemd/system/ssh.service; enabled; preset: enab
   Active: active (running) since Tue 2026-10-13 10:55:59 WIB; 5min ago
   TriggeredBy: • ssh.socket
     Docs: man:sshd(8)
           man:sshd_config(5)
   Process: 796 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 811 (sshd)
     Tasks: 1 (limit: 2263)
    Memory: 2.1M (peak: 2.4M)
       CPU: 31ms
    CGroup: /system.slice/ssh.service
            └─811 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"
```

Gambar 4. Perintah Untuk Memastikan Status SSH

2. Pengujian FPR Tanpa Serangan: Melakukan 4 kali inisiasi *remote login* SSH dari Windows *Host* saat kondisi *server* tidak diserang, lalu memeriksa berkas *fast.log* untuk memverifikasi ketiadaan peringatan palsu (*False Negative* = 0).
3. Pengujian FPR Saat Diserang: Melakukan 4 kali inisiasi *remote login* SSH dari Windows *Host* bersamaan saat Kali Linux meluncurkan simulasi serangan *brute force*, lalu memeriksa berkas *fast.log* untuk memverifikasi ketiadaan peringatan palsu (*False Negative* = 0) terhadap IP Windows *Host*.

```
C:\Users\INFINIX>ssh testuser@192.168.10.20
```

Gambar 5. Perintah Inisiasi Koneksi SSH

```
root@ubuntuserver:/home/testuser# cat /var/log/suricata/fast.log
```

Gambar 6. Perintah Untuk menampilkan isi berkas *log* Suricata

```
(kali㉿kali)-[~/skripsi]
$ hydra -l testuser -P wordlist.txt -t 16 ssh://192.168.10.20 -I
```

Gambar 7. Perintah Untuk Menjalankan Brute Force Menggunakan Hydra

```
root@ubuntuserver:/home/testuser# grep "192.168.10.1" /var/log/suricata/fast.log
1 wc -l_
```

Gambar 8. Perintah Untuk Menghitung Alert Dari IP Windows *Host*

Tahapan Pengujian Suricata

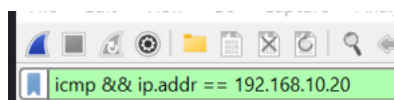
1. Inisiasi & Pengaktifan *Monitoring Baseline* (10 detik): Mengaktifkan utilitas *dstat* pada Ubuntu *Server* untuk merekam penggunaan CPU dan RAM kedalam *file* CSV, *tcpdump* pada Kali Linux untuk merekam transmisi paket serangan ke format *.pcap*, serta Wireshark pada Windows *Host* dan mengirimkan 4 paket ICMP (*ping*) untuk merekam kondisi *baseline* selama 10 detik.

```
root@ubuntuserver:/home/testuser# dstat -c -m --output sken_2-1.csv 1_
```

Gambar 9. Perintah Untuk Mengaktifkan Dstat

```
(root㉿kali)-[/home/kali]
# tcpdump -i eth0 -nn port 22 and host 192.168.10.20 -w sken_2-1.pcap
```

Gambar 10. Perintah Untuk Mengaktifkan Tcpdump



Gambar 11. Filter Pada Wireshark

```
C:\Users\INFINIX>ping 192.168.10.20
```

Gambar 12. Perintah Untuk Melakukan *Ping*

2. Eksekusi simulasi *brute force* dan *ping*: Melancarkan simulasi serangan *brute force* SSH dari Kali Linux menggunakan Instrumen THC-Hydra 9.6 (16 *threads*) dengan memuat kamus kata sandi buatan utilitas Crunch 3.6 (7.776 kombinasi alfanumerik 5 karakter) dan melakukan *continuos ping* dari Windows *Host* menuju *server* untuk mengukur persentase *packet loss*.

```
(kali@kali)-[~/skripsi]
$ hydra -l testuser -P wordlist.txt -t 16 ssh://192.168.10.20 -I
```

Gambar 13. Perintah Untuk Menjalankan *Brute Force* Menggunakan Hydra

```
C:\Users\INFINIX>ping 192.168.10.20 -t
```

Gambar 14. Perintah Untuk Melakukan *Continuos Ping*

3. Penghentian dan Ekspor Data: Menghentikan seluruh instrumen perekaman setelah serangan selesai, mengeksport berkas *log*, CSV *dstat*, berkas *.pcap* untuk dihitung nilai *throughput* serangan, *packet loss*, *detection rate*, dan utilisasi CPU dan RAM.

3. HASIL DAN PEMBAHASAN

Hasil Pengujian *False Positif Rate* (FPR)

Pengujian *False Positive Rate* (FPR) dievaluasi secara terpisah dari pengujian skenario serangan *brute force* untuk mengukur presisi Suricata dalam membedakan aktivitas pengguna sah (*legitimate user*) dari ancaman siber. Pengujian dilakukan dengan melakukan 4 sesi *login* SSH legal dari Windows *Host* (192.168.10.1), baik pada kondisi *server idle* maupun secara simultan saat simulasi serangan *brute force* berlangsung. Sebagaimana dirangkum dalam Tabel 2, hasil pengujian menunjukkan nilai FPR sebesar 0,000% pada seluruh pengujian (*True Negative* = 4, *False Positive* = 0), menunjukkan bahwa aturan Suricata *sid:1000002* memiliki tingkat presisi tinggi tanpa menghasilkan *alert* atau mengganggu akses pengguna sah.

Tabel 2. Hasil Pengujian *False Positive Rate* (FPR)

Kondisi Pengujian	Host Penguji	IP Host	Jumlah Sesi	TN	FP	FPR (%)
Tanpa Serangan	Windows Host	192.168.10.1	4	4	0	0,000%
Tanpa Serangan	Windows Host	192.168.10.1	4	4	0	0,000%

Hasil Evaluasi *Detection Rate* (DR) Suricata

Detection Rate (DR) mengukur persentase keberhasilan sistem deteksi intrusi (Suricata) dalam mengenali inisiasi koneksi anomali yang merupakan bagian dari aktivitas serangan *brute force*. Parameter ini diuji dengan membandingkan jumlah *True Positive* (TP) terhadap total percobaan koneksi aktual yang ditransmisikan oleh instrumen THC-Hydra. Penghitungan TP diformulasikan sebagai jumlah *log* peringatan (*alert*) valid pada Suricata ditambah nilai ambang batas *count* = 10 (TP = Alert + 10). Penambahan nilai *count* = 10 sebagai komponen TP didasarkan pada spesifikasi aturan kustom Suricata

(*detection_filter*: track by_src, count 10, seconds 60;). Batas ambang 10 koneksi ini merupakan jendela evaluasi awal yang disyaratkan oleh aturan untuk memverifikasi pola inisiasi koneksi berfrekuensi tinggi sebelum membangkitkan *alert* pertama. Sementara itu, *False Negative* (FN) dihitung dari selisih antara total percobaan koneksi aktual dan TP (FN = Total Koneksi - TP), yang merepresentasikan inisiasi koneksi serangan yang tidak dicatat sebagai *alert* oleh Suricata. Pada Skenario I (*Baseline*), *server* beroperasi tanpa sistem pengamanan di mana enjin Suricata tidak diaktifkan, sehingga proses deteksi tidak terjadi. Pada Skenario II (Suricata), secara keseluruhan rata-rata tingkat keberhasilan deteksi (*Detection Rate*) tercatat sebesar $86,969 \pm 0,801\%$ dengan rata-rata *True Positive* (TP) sebesar $474,200 \pm 14,805$ dan *False Negative* (FN) sebesar $71 \pm 3,808$ dari rata-rata 545,200 percobaan koneksi. Sebagai contoh representatif pada Iterasi 2, tingkat deteksi Suricata tercatat sebesar 86,957% (470 peringatan dan 10 batas toleransi koneksi awal dari 552 koneksi aktual), yang artinya sekitar 13% (72 inisiasi koneksi) tidak memicu *alert*. Berdasarkan hasil perekaman lalu lintas jaringan, indikator *dropped by kernel* bernilai 0 (nol), yang mengonfirmasi bahwa seluruh paket transmisi berhasil mencapai *server* secara utuh tanpa mengalami pembuangan di tingkat antarmuka jaringan, sebagaimana dirangkum dalam Tabel 3.

Tabel 3. Hasil Pengujian *Detection Rate* (DR) Suricata

Iterasi	Percobaan Koneksi	<i>Alert</i>	<i>Count</i>	TP	FN	DR (%)
1	524	444	10	454	70	86,641
2	552	470	10	480	72	86,957
3	558	483	10	493	65	88,351
4	553	486	10	478	75	86,438
5	539	456	10	466	73	86,456
Rata-rata	$545,200 \pm 13,773$	$464,200 \pm 14,805$	10	$474,200 \pm 14,805$	$71 \pm 3,808$	$86,969 \pm 0,801$

Tingkat deteksi ini berhubungan dengan tingginya pemutusan sesi akibat saturasi pada lapisan aplikasi *server*. Hal ini diobservasi melalui pembedahan muatan paket data (*Deep Packet Inspection*) pada aliran komunikasi TCP menggunakan instrumen Wireshark. Melalui analisis, ditemukan 173 *frame* jaringan yang mentransmisikan pesan "Exceeded MaxStartups". Temuan ini menjadi bukti empiris bahwa layanan *daemon* OpenSSH pada *server* telah mengalami saturasi antrean sesi yang tidak terautentikasi akibat transmisi inisiasi sesi berkecepatan tinggi dari instrumen THC-Hydra (16 *child processes*) yang memicu ambang batas konfigurasi *default* parameter *MaxStartups* OpenSSH (10:30:100), sehingga *server* melakukan penolakan layanan [17].

Reaksi penolakan layanan akibat parameter *MaxStartups* tersebut tervalidasi di lapisan transpor melalui dominasi interupsi koneksi oleh *server*. Berdasarkan analisis arah komunikasi pada instrumen Wireshark, *server* terbukti secara masif mentransmisikan paket interupsi berupa 163 paket berbendera [RST, ACK] dan 181 paket berbendera [RST]. Kondisi ini sejalan dengan temuan 175 anomali pengakhiran sesi pada metrik TCP *Completeness*, yang terdiri dari 163 interupsi paket TCP RST (status 47) dan 12 interupsi ganda FIN serta RST (status 63). Adapun perbedaan minor sejumlah 2 sesi antara

penolakan lapisan aplikasi (173 pesan *server*) dan lapisan transpor (175) tervalidasi secara empiris sebagai akibat dari pembatalan sesi dari IP penyerang (Kali Linux). Interupsi tersebut diproses oleh sistem operasi *server* di lapisan transpor (*Layer 4*) sehingga koneksi terputus sebelum diteruskan ke layanan OpenSSH di lapisan aplikasi (*Layer 7*). Meskipun demikian, inisiasi awal dari 2 sesi tersebut tetap berhasil dideteksi dan dicatat oleh Suricata, membuktikan kapabilitas sensor dalam menginspeksi lalu lintas secara independen di tingkat antarmuka fisik, jauh sebelum pemutusan koneksi oleh sistem operasi *server* dijalankan.

Kendala Suricata dalam mencapai tingkat deteksi 100% pada skenario ini bukanlah akibat dari hilangnya paket di level antarmuka jaringan, melainkan mengindikasikan adanya implikasi dari dinamika pemutusan sesi oleh *server* terhadap mekanisme perakitan aliran data (*tcp-stream reassembly*) Suricata [9]. Pada 72 sesi yang tidak terdeteksi tersebut, pemutusan sesi yang terjadi sebelum proses inspeksi data selesai menyebabkan keterbatasan data yang tersedia bagi mesin deteksi. Sebaliknya, pada 103 sesi lain yang juga mengalami interupsi RST akibat *MaxStartups*, mesin Suricata masih berhasil menghasilkan *alert* sebelum sesi berakhir. Fenomena ini memberikan bukti empiris bahwa nilai *Detection Rate* sebesar 86,969% bukanlah mencerminkan kegagalan atau kelemahan algoritma inspeksi Suricata, melainkan akibat dari aksi pembatasan sesi oleh *daemon* OpenSSH yang memutus paket serangan sebelum mencapai lapisan inspeksi Suricata, sekaligus menegaskan batas operasional Suricata ketika berdiri mandiri tanpa dikombinasikan dengan sistem pencegahan aktif.

Hasil Pengukuran Quality of Service (QoS) Jaringan

Hasil pengukuran metrik QoS jaringan yang meliputi *Throughput* serangan dan *Packet Loss* dirangkum dalam Tabel 4 dan Tabel 5. Pada Skenario I dan Skenario II, *throughput* serangan tercatat masing-masing sebesar $19,677 \pm 1,127$ Kbps dan $19,948 \pm 2,171$ Kbps dengan akumulasi total data serangan mencapai 1,078 MB dan 1,083 MB.

Tabel 4. Hasil Pengukuran *Throughput* Serangan

Skenario Pengujian	Total Data (Byte)	Durasi (s)	<i>Throughput</i> Serangan (Kbps)
Skenario I	$1.130.133,200 \pm 7.724,880$	$460,643 \pm 25,556$	$19,677 \pm 1,127$
Skenario II	$1.136.106,000 \pm 5.818,837$	$459,363 \pm 42,898$	$19,948 \pm 2,171$

Tabel 5. Hasil Pengukuran *Packet Loss*

Skenario Pengujian	Paket Dikirim	Paket Diterima	<i>Packet Loss</i> (%)
Skenario I	$457,800 \pm 26,042$	$457,800 \pm 26,042$	$0,000 \pm 0,000$
Skenario II	$456,000 \pm 44,626$	$455,800 \pm 45,069$	$0,053 \pm 0,119$

Pada Skenario II, teramati sedikit kehilangan paket (*Packet Loss* sebesar $0,053 \pm 0,119\%$). Secara khusus pada Iterasi 3, tercatat hilangnya 1 paket ICMP dari total 377 paket yang ditransmisikan oleh Windows *Host* ($0,265\%$). Hal ini dipicu oleh tingginya aktivitas pemrosesan *Deep Packet Inspection* (DPI) oleh Suricata saat 16 *child process*

Hydra aktif dan beroperasi secara simultan membanjiri antrean antarmuka jaringan *server*.

Hasil Analisis Utilisasi CPU dan RAM

Pengukuran beban komputasi *server* mengevaluasi tingkat penggunaan CPU dan alokasi memori RAM selama simulasi berlangsung. Untuk menjamin keterukuran yang presisi dan mengisolasi beban murni akibat serangan, pengukuran dilakukan pada kondisi *baseline* (sebelum serangan) serta kondisi saat diserang pada setiap iterasi pengujian.

Tabel 6. Hasil Pengukuran Utilisasi CPU

Iterasi	Skenario I (<i>Baseline</i>)			Skenario II (Suricata)		
	Sebelum (%)	Saat (%)	Δ (%)	Sebelum (%)	Saat (%)	Δ (%)
1	11,095	14,356	3,261	5,687	11,913	6,226
2	11,110	13,995	2,885	6,937	11,752	4,816
3	9,460	11,393	1,933	8,408	12,047	3,639
4	8,350	11,502	3,152	7,246	11,217	3,971
5	2,716	6,979	4,264	8,351	11,361	3,010
Rata-rata	$8,546 \pm 3,462$	$11,645 \pm 2,946$	$3,099 \pm 0,835$	$7,326 \pm 1,125$	$11,658 \pm 0,356$	$4,332 \pm 1,243$

Tabel 7. Hasil Pengukuran Utilisasi RAM

Iterasi	Skenario I (<i>Baseline</i>)			Skenario II (Suricata)		
	Sebelum (MB)	Saat (MB)	Δ (MB)	Sebelum (MB)	Saat (MB)	Δ (MB)
1	187,919	213,695	25,776	223,407	249,315	25,908
2	188,240	212,157	23,917	236,334	255,319	18,984
3	188,557	212,486	23,928	241,913	258,942	17,029
4	189,150	216,026	26,876	246,006	251,873	5,867
5	183,254	211,414	28,161	229,635	252,656	23,021
Rata-rata	$187,424 \pm 2,375$	$213,156 \pm 1,803$	$25,732 \pm 1,854$	$235,459 \pm 9,122$	$253,621 \pm 3,665$	$18,162 \pm 7,693$

Berdasarkan data pada Tabel 6 dan Tabel 7, pada Skenario I (tanpa pengamanan), rata-rata kenaikan beban CPU (Δ CPU) tercatat sebesar $3,099 \pm 0,835\%$ dan kenaikan alokasi RAM (Δ RAM) sebesar $25,732 \pm 1,855$ MB. Beban ini merepresentasikan aktivitas *daemon* OpenSSH saat melayani percobaan autentikasi serangan *brute force*. Pada Skenario II (Suricata), rata-rata Δ CPU meningkat menjadi $4,332 \pm 1,243\%$ akibat beban komputasi pemrosesan *Deep Packet Inspection* (DPI) oleh Suricata yang menginspeksi seluruh muatan paket data secara *real-time*. Sebaliknya, rata-rata Δ RAM saat serangan pada Skenario II tercatat sebesar $18,162 \pm 7,693$ MB, di mana kenaikan alokasi memori saat serangan justru lebih efisien dibanding Skenario I karena dipengaruhi oleh alokasi *internal buffer* Suricata *Stream Engine* [9] dan reklamasi memori oleh Kernel Linux yang bekerja secara dinamis [18].

Selain mengukur kenaikan alokasi memori saat serangan, penelitian ini mengukur *overhead* memori statis akibat pengaktifan daemon Suricata saat kondisi *idle* (tanpa serangan). Berdasarkan selisih rata-rata RAM *baseline* antara Skenario II ($235,459 \pm 9,122$ MB) dan Skenario I ($187,424 \pm 2,375$ MB), diperoleh *overhead* memori statis sebesar 48,035 MB. Jika dibandingkan dengan total kapasitas RAM server sebesar 2 GB (2048 MB), *overhead* ini mengkonsumsi 2,345% dari total memori. Tambahan memori statis ini dialokasikan Suricata untuk memuat pustaka *signature*, menginisialisasi modul *inspection engine*, serta menyediakan alokasi awal *streaming buffer*. Hal ini membuktikan bahwa pengoperasian Suricata efisien dan ringan terhadap sumber daya memori *server*.

Matriks Sintesis Terpadu Kinerja Suricata

Seluruh metrik evaluasi dari pengujian Suricata mandiri dirangkum secara komprehensif dalam Matriks Sintesis Terpadu pada Tabel 8.

Tabel 8. Matriks Sintesis Terpadu Kinerja Suricata

Parameter Evaluasi	Skenario I	Skenario II (Suricata)
Status Sistem Keamanan	Tidak Aktif	Suricata Aktif
Percobaan Koneksi SSH	$545,600 \pm 16,288$	$545,200 \pm 13,773$
Detection Rate (DR %)	-	$86,969 \pm 0,801$
False Positif Rate (FPR %)	-	$0,000 \pm 0,000\%$
Throughput Serangan (Kbps)	$19,677 \pm 1,127$	$19,948 \pm 2,171$
Packet Loss (%)	$0,000 \pm 0,000\%$	$0,053 \pm 0,119$
Selisih Beban CPU (Δ CPU %)	$3,099 \pm 0,835$	$4,332 \pm 1,243$
Selisih Beban RAM (Δ RAM MB)	$25,732 \pm 1,854$	$18,162 \pm 7,693$
Overhead Memori Static Idle	48,035 MB	

4. KESIMPULAN

Berdasarkan hasil perancangan, pengujian, dan analisis eksperimental lima iterasi yang telah dilaksanakan, kesimpulan dari penelitian ini adalah sebagai berikut:

1. Suricata berhasil diimplementasikan pada Ubuntu Server 24.04 LTS tanpa modifikasi kode sumber sistem. Suricata terbukti andal menginspeksi aliran data TCP port 22 secara *real-time* melalui rekonstruksi *tcp-stream reassembly* dan aturan berbasis *signature*.
2. Pengujian *False Positive Rate* (FPR) yang dievaluasi secara terpisah menghasilkan nilai 0,000% (*True Negative* = 4, *False Positive* = 0). Aturan kustom Suricata mampu menganalisis jaringan tanpa membangkitkan *alert* palsu pada sesi *login* legal administrator baik dalam kondisi *server idle* maupun saat diserang.
3. Pengujian efektivitas deteksi Suricata menghasilkan *Detection Rate* (DR) rata-rata sebesar $86,969 \pm 0,801\%$. Kondisi DR di bawah 100% tervalidasi secara empiris dipicu oleh penolakan koneksi dari *daemon* OpenSSH akibat terlampauinya batas *MaxStartups* (10:30:100) yang memicu paket TCP RST/RST-ACK, sehingga sebagian paket serangan terputus sebelum Suricata sempat mencatat peringatan.

4. Pengoperasian Suricata memberikan dampak operasional yang terukur pada jaringan dan *server*: *throughput* serangan pada Skenario II tercatat sebesar $19,948 \pm 2,171$ Kbps (dibandingkan $19,677 \pm 1,127$ Kbps pada Skenario I), *packet loss* sebesar $0,053 \pm 0,119\%$ akibat kepadatan inspeksi DPI, selisih kenaikan CPU (Δ CPU) sebesar $4,332 \pm 1,243\%$ (dibandingkan $3,099 \pm 0,835\%$ pada Skenario I), dan selisih kenaikan RAM saat serangan (Δ RAM) sebesar $18,162 \pm 7,693$ MB (dibandingkan $25,732 \pm 1,855$ MB pada Skenario I), dengan *overhead* memori statis *daemon* Suricata saat *idle* sebesar 48,035 MB (2,345% dari kapasitas RAM server 2 GB). Secara umum, Suricata efisien namun membutuhkan sistem mitigasi otomatis aktif untuk mengatasi saturasi sesi di lapisan aplikasi *server*. Penelitian selanjutnya disarankan menguji penerapan Suricata *Inline Mode* (IPS / NFQUEUE) untuk pemblokiran paket langsung di antarmuka jaringan, menguji skenario serangan terdistribusi (*Distributed Brute Force*), serta mengevaluasi variasi aturan deteksi pada protokol terenkripsi lainnya.

UCAPAN TERIMA KASIH

Penulis menyampaikan terima kasih yang sebesar-besarnya kepada Bapak Tengku Mohd. Diansyah, S.Kom., M.Kom. selaku Dosen Pembimbing atas bimbingan, arahan, dan dukungan akademis yang sangat berharga selama proses penyusunan penelitian ini. Artikel ilmiah ini disusun berdasarkan hasil riset mandiri skripsi penulis.

DAFTAR PUSTAKA

- [1] T. Ariyadi, M. A. Nuriansyah, E. Rahmadan, and D. Saputra, "Keamanan FTP Server Berbasis IPS Menggunakan Sistem Operasi Linux Ubuntu Versi 24.04. Jurnal Penelitian Multidisiplin Bangsa," *JURNAL PENELITIAN MULTIDISIPLIN BANGSA*, vol. 2, no. 1, pp. 49–55, Jun. 2025, doi: <https://doi.org/10.59837/jpnmb.v2i1.428>.
- [2] B. R. Utomo, N. H. J. A, A. K. Jati, I. A. Saputro, and Moch. H. Purwidianoro, "Analisis implementasi keamanan jaringan dengan fail2ban terhadap serangan bruteforce," in *SEMINAR NASIONAL AMIKOM SURAKARTA (SEMNAS)*, Nov. 2024, pp. 1211–1223. Accessed: Aug. 08, 2026. [Online]. Available: <https://ojs.amikomsolo.ac.id/index.php/semnasa/article/view/634>
- [3] T. M. Diansyah, I. Faisal, A. Perdana, B. O. Sembiring, and T. H. Sinaga, "Analysis Of Using Firewall And Single Honeypot In Training Attack On Wireless Network," in *Journal of Physics: Conference Series 930 012038*, IOP Publishing, 2017. doi: 10.1088/1742-6596/930/1/012038.
- [4] Farhannullah and M. Hardjianto, "Sistem Monitoring Serangan SSH Dengan Metode Intrusion Prevention System (IPS) Fail2ban Menggunakan Python Pada Sistem Operasi Linux," *Jurnal TICOM: Technology of Information and Communication*, vol. 11, no. 1, pp. 33–38, Sep. 2022, doi: <https://doi.org/10.70309/ticom.v11i1.68>.

-
- [5] P. Balqis and R. Rahman, "Analisis Keamanan Layanan SSH terhadap Brute Force Attack," *Merkurius: Jurnal Riset Sistem Informasi dan Teknik Informatika*, vol. 3, no. 4, pp. 240–246, Jul. 2025, doi: 10.61132/merkurius.v3i4.949.
- [6] M. Ridho, A. Hafizh, I. Dani, and T. Ariyadi, "Peningkatan Keamanan SSH Server Berbasis Linux melalui Implementasi Fail2Ban dan Uji Serangan Brute Force," *Jurnal Penelitian Multidisiplin Bangsa*, vol. 1, no. 12, pp. 2206–2214, 2025, doi: <https://doi.org/10.59837/jpnmb.v1i12.431>.
- [7] B. Raharjo, *Keamanan SISTEM INFORMASI*. Semarang: Yayasan Prima Agus Teknik, 2021.
- [8] M. H. Qutqut, A. Ahmed, M. K. Taqi, J. Abimanyu, E. T. Ajes, and F. Alhaj, "A Comparative Evaluation of Snort and Suricata for Detecting Data Exfiltration Tunnels in Cloud Environments," *Journal of Cybersecurity and Privacy*, vol. 6, no. 1, Feb. 2026, doi: 10.3390/jcp6010017.
- [9] Open Information Security Foundation, "Suricata User Guide," Suricata Documentation. Accessed: May 06, 2026. [Online]. Available: <https://docs.suricata.io/en/suricata-8.0.6/>
- [10] M. Zaini, A. Atthariq, and A. Anwar, "Implementasi Intrusion Detection System Menggunakan Suricata Pada Jaringan Komputer," *Jurnal Teknologi Rekayasa Informasi dan Komputer*, vol. 8, no. 2, pp. 30–38, Sep. 2025, doi: 10.30811/jtrik.v8i2.7440.
- [11] O. Rivaldi and N. L. Marpaung, "Penerapan Sistem Keamanan Jaringan Menggunakan Intrusion Prevention System Berbasis Suricata," *JURNAL INOVTEK POLBENG*, vol. 8, no. 1, pp. 141–153, 2023, Accessed: Aug. 08, 2026. [Online]. Available: <https://pdfs.semanticscholar.org/d427/9b536209c0ddd30d641f3794e6edcbf138a2.pdf>
- [12] A. A. Kurniawan, Jusak, and Musayyanah, "Intrusion Detection System Using Deep Learning for DoS Attack Detection," *JEECS (Journal of Electrical Engineering and Computer Sciences)*, vol. 6, no. 2, pp. 1087–1098, Dec. 2021, doi: 10.54732/jeees.v6i2.203.
- [13] S. Khadafi, D. Y. Pratiwi, and E. Alfianto, "KEAMANAN FTP SERVER BERBASIS IDS DAN IPS MENGGUNAKAN SISTEM OPERASI LINUX UBUNTU," *Jurnal Ilmiah NERO*, vol. 6, no. 1, p. 2021, 2021, Accessed: Aug. 08, 2026. [Online]. Available: https://www.researchgate.net/publication/351031729_KEAMANAN_FTP_SERVER_BERBASIS_IDS_DAN_IPS_MENGGUNAKAN_SISTEM_OPERASI_LINUX_UBUNTU
- [14] T. Ylonen and C. Lonvick, "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol," Jan. 2006. Accessed: Aug. 05, 2026. [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc4253>
- [15] K. B. A. Nurcahyo and A. Prihanto, "ANALISIS QUALITY OF SERVICE (QOS) PADA JARINGAN VLAN (VIRTUAL LOCAL AREA NETWORK)," *Journal of*

- [16] D. Wahyudi, M. Mujiono, and M. Nur Fu'ad, "Analisis Quality of Service (QoS) pada Jaringan Nirkabel di Akademi Komunitas Negeri Putra Sang Fajar Blitar Berdasarkan Standar Parameter TIPHON: Studi Kasus Program Studi Administrasi Server dan Jaringan Komputer," *JAMI: Jurnal Ahli Muda Indonesia*, vol. 5, no. 2, pp. 164–171, Dec. 2024, doi: 10.46510/jami.v5i2.343.
- [17] OpenBSD, "OpenSSH Manual Pages," OpenSSH Manual Pages. Accessed: May 07, 2026. [Online]. Available: <https://www.openssh.org/manual.html>
- [18] The Kernel Development Community, "The Linux Kernel documentation," The Linux Kernel. Accessed: Aug. 04, 2026. [Online]. Available: <https://docs.kernel.org/>