Journal of Applied Informatics and Computing (JAIC)
Vol.x, No.x, Juli 201x, pp. xx~xx
e-ISSN: 2548-6861

ANALISIS PERFORMA WORKFLOW SOC PADA SOFTWARE OPEN SOURCE N8N DAN SHUFFLE YANG TERINTEGRASI REAL TIME DENGAN SIEM DAN THREAT INTELLIGENCE

Sandi Ridho Illahi[1]*, Antoni Haikal[2]*
*** Rekayasa Keamanan Siber, Politeknik Negeri Batam**
**** Politeknik Negeri Batam**
sandi.4332211001@students.polibatam.ac.id [1], antoni@polibatam.ac.id [2]

Article Info

Article history:

Received …
Revised …
Accepted …

Keyword:

n8n, Security Operations Center, Shuffle, SOAR, Workflow Automation.

ABSTRACT

The rapid growth of digital infrastructure increases the complexity of cyber threats, bringing operational challenges to modern Security Operations Centers (SOCs) such as high alert volumes and alert fatigue. Security Orchestration, Automation, and Response (SOAR) addresses this issue, but commercial solutions are often too expensive for Small and Medium Enterprises (SMEs). Open source workflow automation platforms offer a more responsive, stable, and suitable alternative for building automation with an affordable financing scale. This study aims to analyze and compare the performance of two open source platforms, n8n and Shuffle, integrated in real time with Security Information and Event Management (SIEM) and threat intelligence. An experimental quantitative method was used to evaluate performance under two load scenarios: a light load of 100 requests and a heavy load of 1,000 requests. Both scenarios were conducted on the same server environment running Ubuntu Server 22.04 with an Intel Xeon E5540 14 core processor and 16 GB of RAM, using n8n version 2.25.7 and Shuffle version 2.2.0. The evaluated metrics include CPU usage, RAM usage, node latency, total workflow latency, throughput, and tail latencies (p95 and p99). The results indicate that n8n outperformed Shuffle across all evaluated metrics. Under a heavy load of 1000 requests, n8n recorded a lower average CPU usage of 1.96% and RAM usage of 4968.06 MB, compared to Shuffle which reached 16.21% and 9118.83 MB, respectively. Furthermore, n8n demonstrated a significantly lower average total workflow latency of 2.476 seconds compared to Shuffle's 12.94 seconds, alongside a higher throughput capacity. In conclusion, n8n provides more stable and faster automation performance with lower resource consumption, making it highly suitable for fulfilling SOC automation requirements in SME environments.

I. PENDAHULUAN

Perkembangan teknologi digital telah meningkatkan ketergantungan organisasi terhadap sistem informasi, layanan berbasis jaringan, dan infrastruktur digital yang saling terhubung. Di sisi lain, kondisi tersebut juga memperluas permukaan serangan dan meningkatkan kompleksitas ancaman siber [1]. *Security Operations Center* (SOC) berperan sebagai pusat operasi keamanan yang bertugas memantau, menganalisis, dan merespons insiden keamanan [2]. Namun, SOC modern menghadapi tantangan operasional seperti tingginya volume *alert*, munculnya *false positive*, keterbatasan sumber daya analis, serta beban kerja berulang yang berisiko menyebabkan *alert fatigue* [1], [3], [4].

Untuk mengatasi permasalahan tersebut, pendekatan *Security Orchestration, Automation, and Response* (SOAR) digunakan untuk mengintegrasikan berbagai perangkat keamanan dan mengotomatisasi *workflow* respons insiden [5], [6]. Bagi kalangan *Small and Medium Enterprises* (SME),

e-ISSN: 2548-6861

adopsi solusi SOAR komersial sering kali terkendala oleh tingginya biaya dan kompleksitas implementasi. Oleh karena itu, platform *workflow automation open source* hadir sebagai alternatif yang relevan karena dapat membantu membangun otomasi SOC dengan skala pembiayaan yang lebih terjangkau [7]. Solusi keamanan berbasis *open source* dinilai esensial bagi SME karena mampu menyediakan kemampuan *monitoring* dan respons keamanan dengan biaya yang lebih terjangkau dibandingkan solusi *proprietary* [8].

n8n dan Shuffle merupakan dua platform *workflow automation open source* yang dapat diandalkan untuk membangun otomasi SOC berbasis *webhook* dan *Application Programming Interface* (API) [9]. Keduanya dapat diintegrasikan secara langsung dengan Wazuh sebagai *Security Information and Event Management* (SIEM) [10], *Malware Information Sharing Platform* (MISP) sebagai *threat intelligence platform*, Wazuh *Active Response* sebagai mekanisme respons, serta Telegram sebagai media notifikasi. Penelitian sebelumnya oleh Okroshidze [11] telah membahas perbandingan n8n dan Shuffle sebagai opsi otomasi workflow SOC untuk SME. Akan tetapi, penelitian tersebut hanya berfokus pada aspek perbandingan fitur, kemudahan integrasi, dan dampak operasional, tanpa melakukan pengukuran performa teknis secara kuantitatif. Akibatnya, belum tersedia data empiris yang dapat digunakan oleh organisasi SME untuk menentukan platform mana yang lebih efisien dari sisi konsumsi sumber daya dan waktu respons ketika menjalankan workflow SOC yang identik.

Berdasarkan *research gap* tersebut, penelitian ini bertujuan untuk menganalisis dan membandingkan performa *workflow* SOC pada platform n8n versi 2.25.7 dan Shuffle versi 2.2.0 yang terintegrasi secara *real time* dengan SIEM dan T*hreat Intelligence*. Dengan menyediakan data performa teknis yang terukur, penelitian ini diharapkan dapat membantu organisasi SME dalam memilih platform *workflow automation* yang paling sesuai dengan keterbatasan sumber daya komputasi dan anggaran yang dimiliki. Pengujian dilakukan melalui dua skenario pembebanan, yaitu 100 *request* untuk merepresentasikan beban ringan dan 1000 *request* untuk beban tinggi. Metrik performa yang dievaluasi mencakup *Central Processing Unit* (*CPU) usage*, *Random Access Memory* (*RAM) usage*, *latency* per *node*, total *latency workflow*, *latency* rata-rata, *latency* maksimum, *throughput*, *tail latency* p95, dan *tail latency* p99. Hasil penelitian ini diharapkan dapat memberikan gambaran empiris yang komprehensif mengenai platform *workflow automation open source* yang lebih responsif, stabil, dan paling sesuai untuk memenuhi kebutuhan otomasi SOC pada lingkungan SME.

II. METODE

Penelitian ini menggunakan metode kuantitatif eksperimental untuk membandingkan performa *workflow* SOC pada dua platform *workflow automation open source*, yaitu n8n dan Shuffle. Metode ini digunakan karena penelitian berfokus pada pengukuran metrik numerik yang dapat dianalisis secara kuantitatif, seperti CPU *usage*, RAM *usage, latency, throughput, tail latency* p95, dan *tail latency* p99. Evaluasi performa sistem perlu dilakukan dengan lingkungan yang terkontrol, skenario pengujian yang konsisten, dan metrik yang jelas agar hasil yang diperoleh dapat dibandingkan secara objektif [12].

Workflow SOC yang diuji pada penelitian ini diimplementasikan secara identik pada n8n dan Shuffle. Workflow tersebut terintegrasi dengan Wazuh sebagai *Security Information and Event Management* (SIEM), *Malware Information Sharing Platform* (MISP) sebagai Threat Intelligence platform, Wazuh *Active Response* sebagai mekanisme respons, dan Telegram sebagai media notifikasi. Alur *workflow* yang digunakan terdiri dari Wazuh *Alert*, API *Token* Wazuh, Wazuh *Active Response*, pencatatan *Latency Active Response*, MISP *Check Internet Protocol* (IP), pencatatan *Latency* MISP, Telegram Notification, pencatatan *Latency* Telegram, dan *Finished*.

Implementasi *workflow* SOC pada n8n dan Shuffle dirancang dengan alur yang identik agar perbedaan hasil pengujian lebih merepresentasikan karakteristik performa masing-masing platform. *Workflow* n8n ditunjukkan pada Gambar 1, sedangkan implementasi *workflow* yang sama pada platform Shuffle ditunjukkan pada Gambar 2.

Gambar 1. Workflow n8n

Gambar 2. Workflow Shuffle

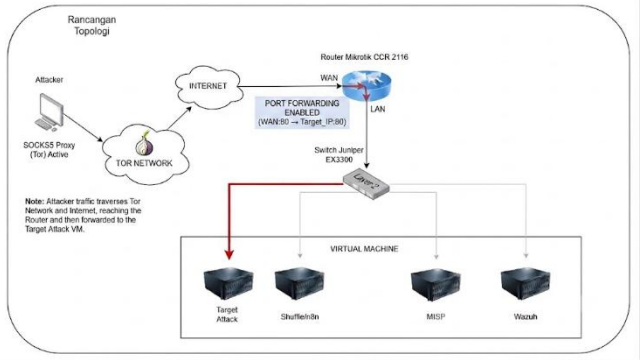
Kesamaan alur *workflow* diperlukan agar perbedaan hasil pengujian lebih merepresentasikan karakteristik performa masing-masing platform, bukan disebabkan oleh perbedaan desain proses.

Pengujian dilakukan pada lingkungan server berbasis Ubuntu Server 22.04 dengan spesifikasi prosesor Intel(R) Xeon(R) CPU E5540 @ 2.53 GHz 14 core, RAM 16 GB, dan Harddisk 500 GB. Platform n8n versi 2.25.7 dan Shuffle versi 2.2.0 dijalankan pada lingkungan server yang sama dengan alokasi sumber daya yang identik agar perbandingan performa dilakukan secara konsisten dan adil. Kedua platform diuji secara bergantian bukan bersamaan untuk memastikan tidak ada interferensi penggunaan sumber daya antar platform selama proses pengujian berlangsung. Penggunaan lingkungan yang sama bertujuan untuk memastikan bahwa perbedaan hasil pengujian lebih merepresentasikan karakteristik performa n8n dan Shuffle, bukan disebabkan oleh perbedaan sumber daya komputasi dan dapat dilihat pada Tabel I berikut.

TABEL I
SPESIFIKASI LINGKUNGAN PENGUJIAN

Komponen	Spesifikasi
Sistem operasi	Ubuntu Server 22.04
Prosesor	Intel(R) Xeon(R) CPU E5540 @ 2.53 GHz 14 Core
Random Access Memory (RAM)	16 GB
Hard Disk Drive	500 GB
Tools monitoring resource	System Activity Report (SAR)
Platform yang diuji	n8n v2.25.7 dan Shuffle v2.2.0
Komponen integrasi	Wazuh, MISP, Wazuh Active Response, Telegram

Lingkungan pengujian terdiri dari empat komponen utama yang terhubung melalui switch Layer 2, yaitu *server* target serangan, *server* Shuffle/n8n, *server* MISP, dan *server* Wazuh. Rancangan topologi jaringan pengujian ditunjukkan pada Gambar 3.



Gambar 3. Topologi Fisik

Variabel dalam penelitian ini terdiri dari variabel independen, variabel dependen, dan variabel kontrol. Variabel independen adalah platform *workflow automation* yang diuji, yaitu n8n dan Shuffle. Variabel dependen adalah metrik performa yang terdiri dari CPU *usage*, RAM *usage*, *latency* per *node*, total *latency workflow*, *latency* rata-rata, *latency* maksimum, *throughput*, *tail latency* p95, dan *tail latency* p99. Variabel kontrol meliputi lingkungan server, sistem operasi, jumlah *core* CPU, kapasitas RAM, dataset, jumlah *request*, pola pengiriman *request*, *timeout*, layanan eksternal, dan format pencatatan log/CSV dan dapat dilihat pada Tabel II.

TABEL II
VARIABEL PENELITIAN

Jenis Variabel	Nama Variabel	Indikator / Parameter
Variabel independent	Platform workflow automation	n8n dan Shuffle
Variabel dependen	Performa workflow SOC	CPU usage, RAM usage, latency per node, total latency workflow, latency rata-rata, latency maksimum, throughput, p95, dan p99
Variabel kontrol	Kondisi eksperimen	Server, sistem operasi, alokasi CPU, RAM, dataset, jumlah request, pola pengiriman request, timeout, layanan eksternal, dan format log/CSV

Simulasi beban dilakukan dengan menghasilkan *event* keamanan yang dikirimkan ke *server* target untuk memicu alert pada Wazuh. Request dikirim melalui *proxy SOCKS* berbasis *Tor* untuk memperoleh variasi alamat IP publik yang berpotensi memiliki reputasi berisiko pada sistem *Threat Intelligence*. Untuk memicu alert keamanan yang bervariasi pada Wazuh, pengiriman *request* dimodifikasi secara dinamis menggunakan *script* automasi berbasis *Python*. *Script* tersebut dirancang untuk merotasi tiga jenis payload serangan web secara acak pada setiap iterasi pengiriman. *Payload* disisipkan melalui modifikasi parameter URI (*Uniform Resource Identifier*) *Path* dan *header User-Agent* pada *HTTP GET request*. Detail dari *payload* yang disimulasikan dapat dilihat pada Tabel III.

TABEL III
DETAIL PAYLOAD SIMULASI SERANGAN

Jenis Serangan	Struktur Payload
Shellshock	() { :; }; /bin/cat /etc/passwd #lab-{n}
Cross-Site Scripting (XSS)	/?q=<script>alert('wazuh-xss-{n}')</script>
Path Traversal	/?file=../../../../etc/passwd&lab={n}

Simulasi ini dilakukan dalam lingkungan pengujian terkontrol dan terotorisasi, serta hanya diarahkan ke *server* target yang telah disiapkan untuk kebutuhan eksperimen. *Event* yang diterima Wazuh kemudian memicu *workflow* pada n8n dan Shuffle untuk menjalankan proses *active response*, pengecekan *Threat Intelligence* melalui MISP, pengiriman notifikasi, dan pencatatan *latency*.

Desain beban uji dibagi menjadi dua skenario, yaitu beban ringan dan beban tinggi. Skenario beban ringan menggunakan 115 *payload* untuk memperoleh 100 *request* yang dianalisis, sedangkan skenario beban tinggi menggunakan 1150 payload untuk memperoleh 1000 *request* yang dianalisis. Jumlah *payload* dibuat lebih besar dari jumlah request yang dianalisis untuk memastikan data valid tetap mencukupi selama proses pengujian dan pencatatan workflow. Kedua platform diuji menggunakan mekanisme simulasi, dataset, pola pengiriman *request, timeout*, lingkungan *server*, dan format pencatatan yang sama.

TABEL IV
DESAIN BEBAN UJI

Skenario	Request Dianalisis	Tujuan	Platform

e-ISSN: 2548-6861

Beban ringan	100 request	Mengukur performa workflow pada beban rendah	n8n dan Shuffle
Beban tinggi	1000 request	Mengukur performa workflow pada beban tinggi	n8n dan Shuffle

Pengukuran performa dilakukan dengan mencatat waktu eksekusi pada setiap tahapan utama *workflow*. *Latency* per *node* dihitung dari selisih waktu selesai dan waktu mulai pada *node* tertentu, sedangkan total *latency workflow* dihitung dari selisih waktu selesai *workflow* dan waktu mulai *workflow*. *Throughput* dihitung berdasarkan jumlah *request* yang diproses dalam durasi pengujian. CPU *usage* dan RAM *usage* diperoleh dari hasil *monitoring resource* menggunakan *System Activity Report* (SAR) pada sistem operasi *Linux* [13]. SAR digunakan untuk mencatat penggunaan sumber daya sistem selama proses pengujian berlangsung, sehingga data *resource* yang diperoleh dapat dianalisis berdasarkan interval waktu pengujian.

Pemilihan metrik pengujian pada penelitian ini disesuaikan dengan kebutuhan evaluasi performa *workflow* SOC. CPU *usage* dan RAM *usage* digunakan untuk mengetahui konsumsi sumber daya sistem selama *workflow* dijalankan. *Latency* digunakan untuk mengukur waktu respons pada setiap tahapan *workflow*, sedangkan *throughput* digunakan untuk melihat kemampuan platform dalam memproses *request* per satuan waktu [14]. Selain *latency* rata-rata dan *latency* maksimum, penelitian ini juga menggunakan tail *latency* p95 dan p99 untuk mengetahui kestabilan waktu eksekusi, karena sebagian kecil *request* dengan waktu proses tinggi dapat memengaruhi performa sistem secara keseluruhan [15].

TABEL V
DEFINISI METRIK PENGUJIAN

No	Metrik	Definisi	Rumus
1	CPU usage	Persentase penggunaan CPU selama pengujian	Rata-rata penggunaan CPU selama interval pengujian
2	RAM usage	Penggunaan memori selama pengujian	Rata-rata penggunaan RAM selama interval pengujian
3	Total latency workflow	Waktu total dari awal workflow hingga selesai	Waktu selesai workflow sampai waktu mulai workflow
4	Latency rata-rata	Rata-rata waktu eksekusi request	Total latency seluruh request / jumlah request
5	Latency maksimum	Nilai latency paling besar selama pengujian	Nilai latency terbesar
6	P95 latency	Nilai latency pada persentil ke-95	95% request memiliki latency ≤ nilai p95
7	P99 latency	Nilai latency pada persentil ke-99	99% request memiliki latency ≤ nilai p99
8	Throughput	Jumlah request yang diproses per satuan waktu	Jumlah request / total durasi pengujian
9	Latency per node	Waktu yang dibutuhkan satu node untuk menyelesaikan proses	Waktu selesai node sampai waktu mulai node

Data penelitian dikumpulkan dari *log* eksekusi *workflow*, *file* CSV *latency*, hasil *monitoring resource* SAR, dan durasi total pengujian. Data *latency* diperoleh dari pencatatan timestamp pada setiap tahapan utama *workflow*, sedangkan data CPU dan RAM diperoleh dari hasil *monitoring* sistem selama pengujian berlangsung. Seluruh data dianalisis secara kuantitatif dengan membandingkan hasil n8n dan Shuffle pada masing-masing skenario pengujian.

Tahapan analisis dilakukan dengan menghitung nilai *latency* per *node*, total *latency workflow*, *latency* rata-rata, *latency* maksimum, *throughput*, p95, p99, CPU *usage*, dan RAM *usage*. Hasil pengukuran kemudian dibandingkan antara n8n dan Shuffle pada skenario beban ringan dan beban tinggi. Platform dengan penggunaan CPU dan RAM lebih rendah dinilai lebih baik dari sisi sumber daya, sedangkan platform dengan *latency* rata-rata, *latency* maksimum, p95, dan p99 lebih rendah dinilai lebih stabil dari sisi waktu eksekusi. Sementara itu, nilai *throughput* digunakan untuk melihat kemampuan platform dalam memproses *request* per satuan waktu.

III. HASIL DAN PEMBAHASAN

Pengujian performa dilakukan untuk membandingkan *workflow* SOC pada platform n8n dan Shuffle. Pengujian dilakukan menggunakan dua skenario beban, yaitu beban ringan dengan 100 request yang dianalisis dan beban tinggi dengan 1000 request yang dianalisis. Data yang digunakan dalam analisis diperoleh dari *log latency workflow*, hasil *monitoring resource* menggunakan *System Activity Report* (SAR), serta durasi eksekusi *workflow* pada masing-masing platform.

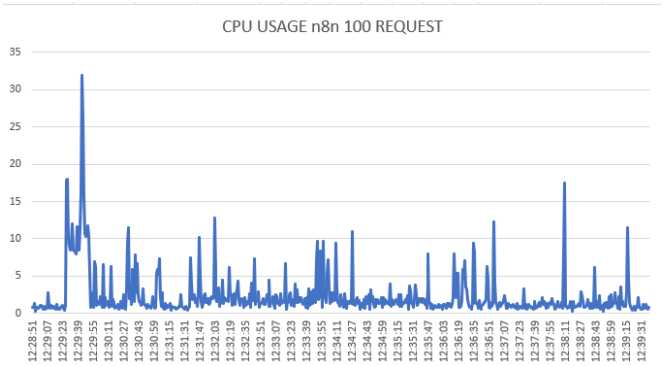
Simulasi *event* keamanan digunakan untuk memicu *alert* pada Wazuh sehingga *workflow* SOC pada n8n dan Shuffle dapat berjalan sesuai alur yang telah dirancang. *Request* dikirim melalui *proxy SOCKS* berbasis *Tor* untuk memperoleh variasi alamat IP publik yang berpotensi memiliki reputasi berisiko pada sistem Threat Intelligence. Kedua platform diuji menggunakan mekanisme simulasi, lingkungan server, pola pengiriman *request*, dan format pencatatan data yang sama agar hasil perbandingan dapat dianalisis secara objektif.

A. Hasil Pengujian CPU *Usage*

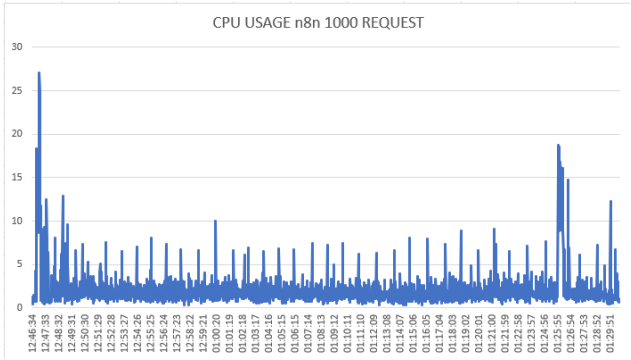
Pengujian CPU *usage* dilakukan untuk mengetahui tingkat penggunaan prosesor selama *workflow* SOC dijalankan. Nilai CPU diperoleh dari hasil monitoring SAR selama proses pengujian berlangsung. Hasil pengujian CPU *usage* pada n8n dan Shuffle ditampilkan pada Tabel VI.

TABEL VI
HASIL PENGUJIAN CPU USAGE

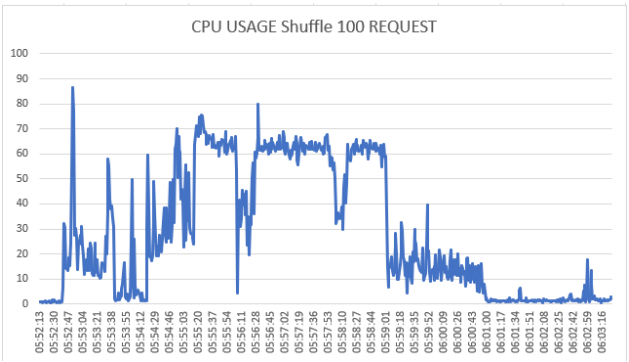
Platform	Skenario	Rata-rata CPU Usage (%)	CPU Maksimum (%)
n8n	100 request	2,30	31,9
n8n	1000 request	1,96	27,10
Shuffle	100 request	28,24	86,54
Shuffle	1000 request	16,21	84,02



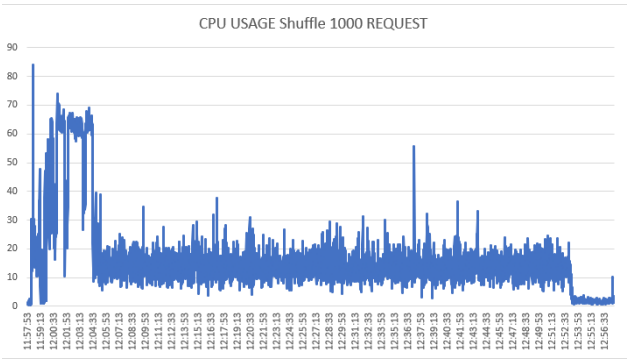
Gambar 4. CPU Usage n8n 100 Request



Gambar 5. CPU Usage n8n 1000 Request



Gambar 6. CPU Usage Shuffle 100 Request



Gambar 7. CPU Usage Shuffle 1000 Request

Berdasarkan Tabel VI, Gambar 4 sampai Gambar 7, n8n menunjukkan penggunaan CPU yang lebih rendah dibandingkan Shuffle pada kedua skenario pengujian. Pada skenario 100 *request*, n8n memiliki rata-rata CPU *usage* sebesar 2,30%, sedangkan Shuffle mencapai 28,24%. Pada skenario 1000 *request*, n8n memiliki rata-rata CPU *usage* sebesar 1,96%, sedangkan Shuffle sebesar 16,21%.

Dari sisi CPU maksimum, Shuffle juga menunjukkan nilai yang jauh lebih tinggi, yaitu 86,54% pada skenario 100 request dan 84,02% pada skenario 1000 *request*. Sementara itu, n8n hanya mencapai CPU maksimum sebesar 31,90% pada skenario 100 *request* dan 27,10% pada skenario 1000 *request*. Hasil ini menunjukkan bahwa n8n lebih ringan dalam penggunaan prosesor, sedangkan Shuffle membutuhkan beban CPU yang lebih besar untuk menjalankan workflow SOC yang sama.

Selain itu, pola pada grafik juga menunjukkan perbedaan karakteristik penggunaan CPU antar platform. Pada n8n, penggunaan CPU cenderung berada pada level rendah dengan beberapa lonjakan sesaat selama *workflow* berjalan. Sebaliknya, pada Shuffle terlihat adanya periode penggunaan CPU yang tinggi dan relatif lebih lama, terutama pada skenario 100 *request*, sehingga menunjukkan beban komputasi yang lebih besar selama proses eksekusi *workflow*.

B. Hasil Pengujian RAM *Usage*

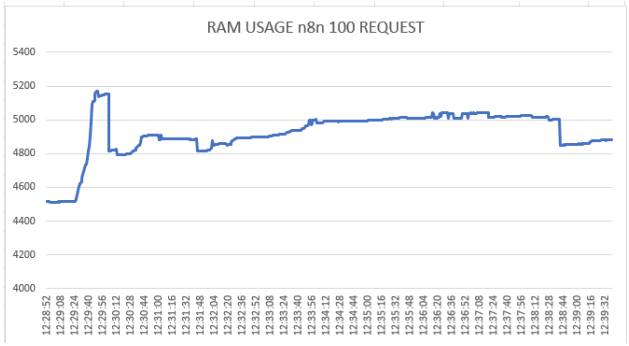
Pengujian RAM *usage* dilakukan untuk mengetahui penggunaan memori selama proses pengujian berlangsung. Nilai RAM diperoleh dari hasil *monitoring* SAR pada interval

e-ISSN: 2548-6861

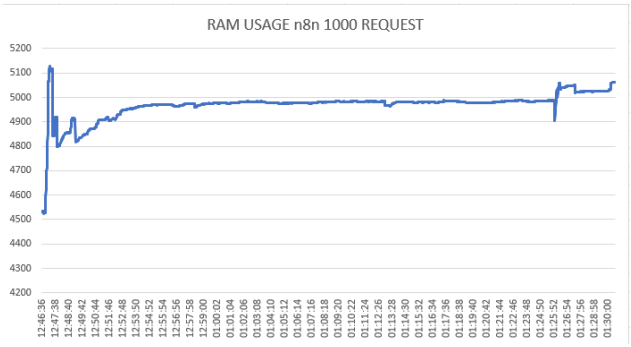
pengujian masing-masing platform. Hasil pengujian RAM *usage* ditampilkan pada Tabel VII.

TABEL VII
HASIL PENGUJIAN RAM USAGE

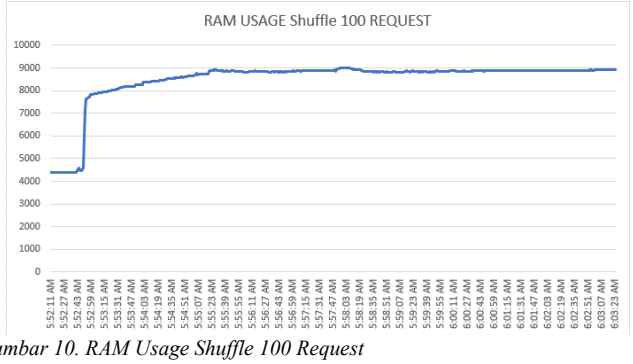
Platform	Skenario	Rata-rata RAM Usage (MB)	RAM Maksimum (MB)
n8n	100 request	4.920,37	5.172,29
n8n	1000 request	4.968,06	5.128,84
Shuffle	100 request	8.477,44	9.015,10
Shuffle	1000 request	9.118,83	9.397,68



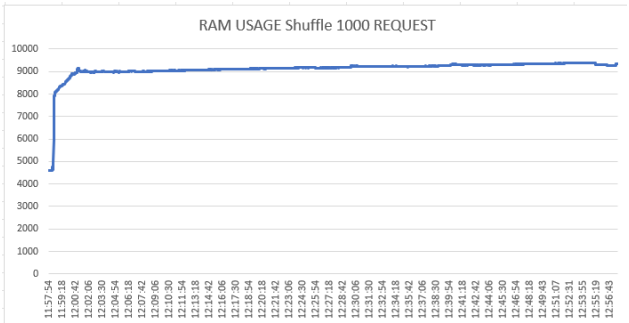
Gambar 8. RAM Usage n8n 100 Request



Gambar 9. RAM Usage n8n 1000 Request



Gambar 10. RAM Usage Shuffle 100 Request



Gambar 11. RAM Usage *Shuffle 1000 Request*

Berdasarkan Tabel VII, Gambar 8 sampai Gambar 11, n8n memiliki penggunaan RAM yang lebih rendah dan lebih stabil dibandingkan Shuffle pada kedua skenario pengujian. Pada skenario 100 request, n8n menggunakan RAM rata-rata sebesar 4.920,37 MB dengan nilai RAM maksimum sebesar 5.172,29 MB. Grafik RAM USAGE n8n 100 *Request* memperlihatkan adanya lonjakan memori di awal pengujian yang kemudian stabil pada kisaran 4.800–5.050 MB. Pada skenario 1.000 *request*, penggunaan RAM n8n sebesar 4.968,06 MB dengan nilai RAM maksimum sebesar 5.128,84 MB, dengan pola stabilisasi yang serupa.

Peningkatan beban dari 100 *request* ke 1.000 *request* hanya mengakibatkan kenaikan RAM rata-rata sebesar 47,69 MB pada n8n, yang setara dengan peningkatan sekitar 0,97%. Hasil ini menunjukkan bahwa n8n memiliki manajemen memori yang sangat baik dan tidak mengalami akumulasi memori yang signifikan meskipun volume request ditingkatkan sepuluh kali lipat.

Shuffle menunjukkan konsumsi memori yang jauh lebih tinggi dibandingkan n8n. Pada skenario 100 *request*, Shuffle menggunakan RAM rata-rata sebesar 8.477,44 MB dengan nilai RAM maksimum sebesar 9.015,10 MB. Pada skenario 1.000 *request*, Shuffle menggunakan RAM rata-rata sebesar 9.118,83 MB dengan nilai RAM maksimum sebesar 9.397,68 MB. Peningkatan RAM dari skenario 100 *request* ke 1.000 *request* pada Shuffle sebesar 641,39 MB atau sekitar 7,57%, yang jauh lebih besar dibandingkan n8n. Dengan demikian, n8n menggunakan memori sekitar 45,9% lebih rendah dibandingkan Shuffle pada skenario 100 *request*, dan 45,5% lebih rendah pada skenario 1.000 *request*. Hasil ini menegaskan bahwa n8n lebih hemat memori dan lebih sesuai untuk lingkungan dengan keterbatasan sumber daya.

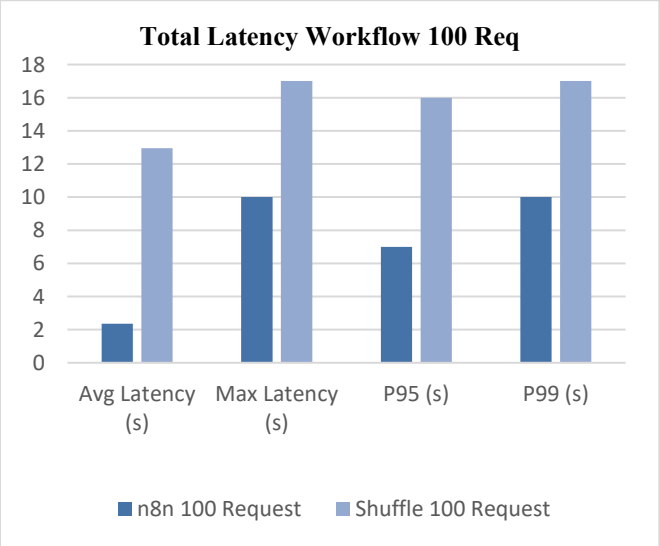
C. Hasil Pengujian Total *Latency Workflow* dan *Throughput*

Total *latency workflow* digunakan untuk mengukur waktu yang dibutuhkan platform dalam menyelesaikan satu proses *workflow* dari awal hingga akhir. Metrik *latency* yang dianalisis meliputi *latency* rata-rata, *latency* maksimum, p95, dan p99. Data *latency* yang dianalisis berjumlah tepat 100 record pada masing-masing platform untuk skenario 100 request, dan 1.000 record untuk skenario 1.000 request. Seluruh eksekusi pada kedua platform tercatat berstatus

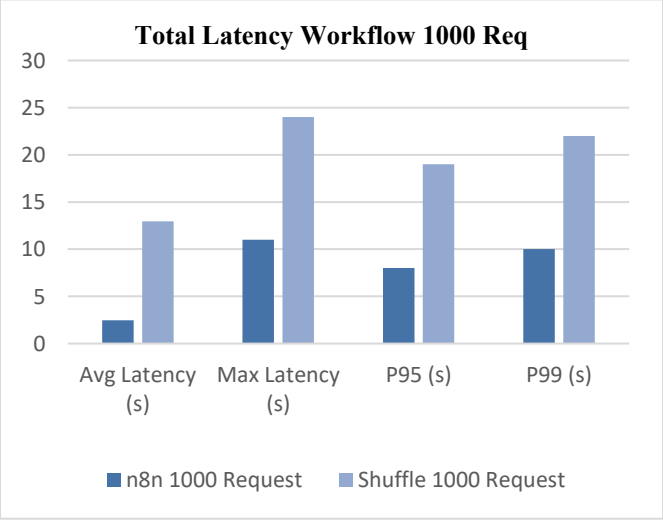
berhasil dengan tingkat keberhasilan 100%. Hasil pengujian total latency workflow ditampilkan pada Tabel VIII.

TABEL VIII
HASIL PENGUJIAN TOTAL LATENCY WORKFLOW

Platform	Skenario	Avg Latency (s)	Max Latency (s)	P95 (s)	P99 (s)
n8n	100 request	2,36	10	7	10
n8n	1000 request	2,476	11	8	10
Shuffle	100 request	12,95	17	16	17
Shuffle	1000 request	12,94	24	19	22



Gambar 12 Total latency workflow 100 request



Gambar 13 Total latency workflow 1000 request

Berdasarkan Tabel VIII, Gambar 12 dan Gambar 13, n8n menunjukkan total *latency workflow* yang lebih rendah dibandingkan Shuffle pada kedua skenario pengujian. Pada skenario 100 *request*, n8n menghasilkan rata-rata *latency* sebesar 2,36 detik, sedangkan Shuffle mencapai 12,95 detik. Pada skenario 1000 *request*, rata-rata latency n8n sebesar 2,476 detik, sementara Shuffle sebesar 12,94 detik.

Dari sisi *latency* maksimum, n8n juga menunjukkan performa yang lebih baik dengan nilai maksimum sebesar 10 detik pada skenario 100 *request* dan 11 detik pada skenario 1000 *request*. Sebaliknya, Shuffle mencapai *latency* maksimum sebesar 17 detik pada skenario 100 *request* dan meningkat menjadi 24 detik pada skenario 1000 *request*.

Nilai *tail latency* menunjukkan pola yang serupa. Pada skenario 100 *request*, n8n memiliki nilai p95 sebesar 7 detik dan p99 sebesar 10 detik, sedangkan Shuffle memiliki nilai p95 sebesar 16 detik dan p99 sebesar 17 detik. Pada skenario 1000 *request*, nilai p95 dan p99 n8n masing-masing sebesar 8 detik dan 10 detik, sedangkan Shuffle mencapai 19 detik dan 22 detik. Hasil ini menunjukkan bahwa n8n mampu mempertahankan waktu respons yang lebih stabil dibandingkan Shuffle, baik pada kondisi beban ringan maupun beban tinggi.

Throughput digunakan untuk mengetahui kemampuan platform dalam memproses *request* per satuan waktu. Hasil pengujian throughput ditampilkan pada Tabel IX.

TABEL IX
HASIL PENGUJIAN THROUGHPUT

Troughput	n8n	Shuffle
100 Request (Request per second	0,423728814	0,266666667
1000 Request (Request per second)	0,423370025	0,341997264
100 Request (Request per minute)	25,42372881	16
1000 Request (Request per minute)	25,40220152	20,51983584

Berdasarkan Tabel VIII, n8n menghasilkan *throughput* yang lebih tinggi dibandingkan Shuffle pada kedua skenario pengujian. Pada skenario 100 *request*, *throughput* n8n mencapai 0,4237 *request* per detik atau setara dengan 25,42 *request* per menit, sedangkan Shuffle mencapai 0,2667 *request* per detik atau 16 *request* per menit.

Pada skenario 1000 *request*, *throughput* n8n sebesar 0,4234 *request* per detik atau 25,40 *request* per menit, sedangkan Shuffle sebesar 0,3420 *request* per detik atau 20,52 *request* per menit. Hasil ini menunjukkan bahwa n8n mampu memproses request lebih banyak dalam satuan waktu yang sama dibandingkan Shuffle.

Konsistensi nilai *throughput* n8n pada kedua skenario juga menunjukkan bahwa peningkatan jumlah *request* tidak

e-ISSN: 2548-6861

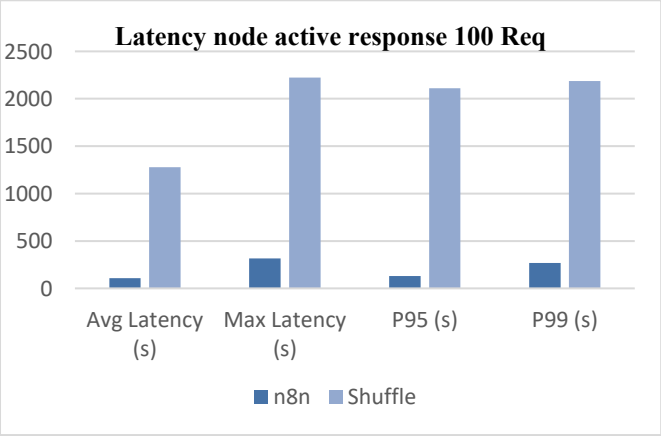
memberikan dampak yang signifikan terhadap kemampuan platform dalam memproses *workflow*. Sebaliknya, meskipun *throughput* Shuffle meningkat pada skenario 1000 *request*, nilainya masih berada di bawah *throughput* yang dihasilkan oleh n8n.

D. Hasil Pengujian *Latency* per *Node*

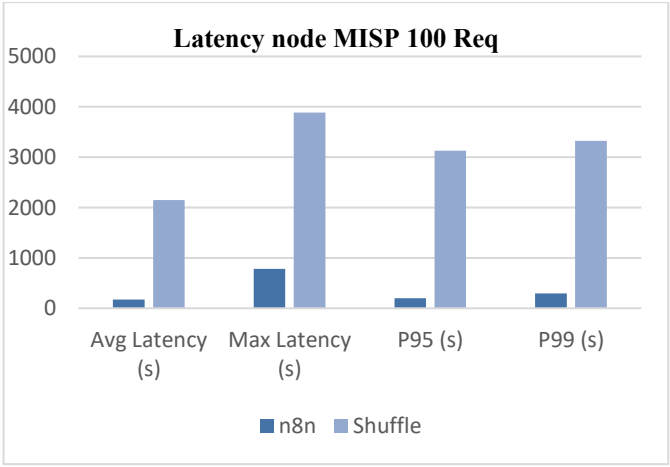
Latency per *node* digunakan untuk mengetahui tahapan *workflow* yang memberikan kontribusi terbesar terhadap total *latency*. Pada penelitian ini, *latency* per *node* dihitung pada tiga tahapan utama, yaitu Wazuh *Active Response*, MISP *Check* IP, dan Telegram *Notification*.

TABEL X
HASIL LATENCY PER NODE PADA SKENARIO 100 REQUEST

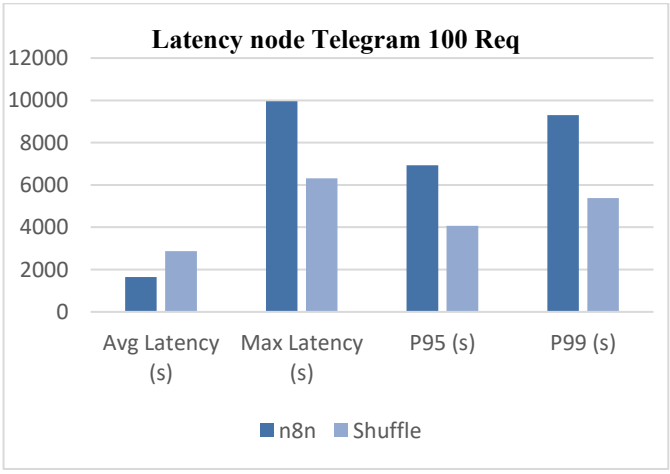
Platform	Node	Avg Latency (ms)	Max Latency (ms)	P95 (ms)	P99 (ms)
n8n	Active Response	108,8	317	131	267
n8n	MISP Check IP	172,38	782	202	297
n8n	Telegram Notification	1641,91	9961	6937	9297
Shuffle	Active Response	1277,72	2224	2112	2187
Shuffle	MISP Check IP	2148,79	3885	3130	3323
Shuffle	Telegram Notification	2876,06	6309	4070	5378



Gambar 14 Latency node active response 100 request



Gambar 15 Latency node MISP 100 request



Gambar 16 Latency node Telegram 100 request

Berdasarkan Tabel X dan Gambar 14 sampai Gambar 16, *node* Telegram *Notification* merupakan tahapan dengan *latency* rata-rata tertinggi pada kedua platform. Pada n8n, rata-rata latency Telegram *Notification* sebesar 1.641,91 ms, sedangkan pada Shuffle mencapai 2.876,06 ms. Perbedaan ini menunjukkan bahwa proses pengiriman notifikasi pada Shuffle membutuhkan waktu yang lebih lama dibandingkan n8n untuk *workflow* yang sama.

Pada *node Active Response*, n8n menghasilkan rata-rata *latency* sebesar 108,80 ms, sedangkan Shuffle mencapai 1.277,72 ms. Hal ini menunjukkan bahwa proses eksekusi respons otomatis pada Shuffle membutuhkan waktu hampir dua belas kali lebih besar dibandingkan n8n. Perbedaan serupa juga terlihat pada *node* MISP *Check* IP, di mana n8n menghasilkan rata-rata *latency* sebesar 172,38 ms, sementara Shuffle mencapai 2.148,79 ms.

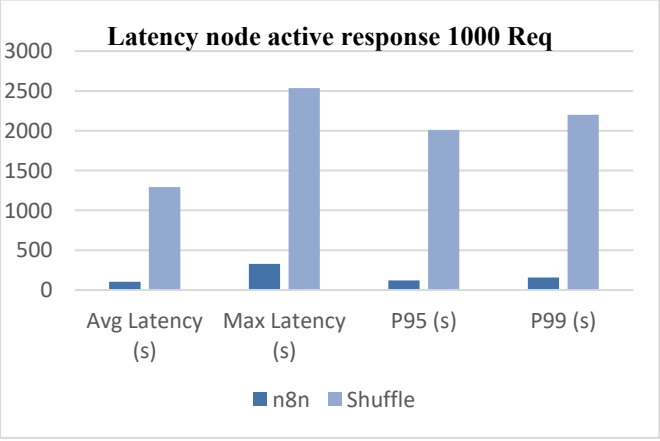
Dari sisi *tail latency*, nilai p95 dan p99 menunjukkan adanya variasi waktu respons pada kedua platform. Pada n8n, node Telegram *Notification* memiliki nilai p95 sebesar 6.937 ms dan p99 sebesar 9.297 ms, sedangkan pada Shuffle nilai p95 mencapai 4.070 ms dan p99 sebesar 5.378 ms. Meskipun

nilai *tail latency* Telegram pada n8n lebih tinggi, rata-rata *latency node* Telegram tetap lebih rendah dibandingkan Shuffle. Hal ini menunjukkan bahwa sebagian besar *request* pada n8n diproses lebih cepat, namun terdapat sejumlah kecil *request* yang mengalami waktu respons lebih tinggi akibat faktor eksternal seperti kondisi jaringan atau respons API Telegram.

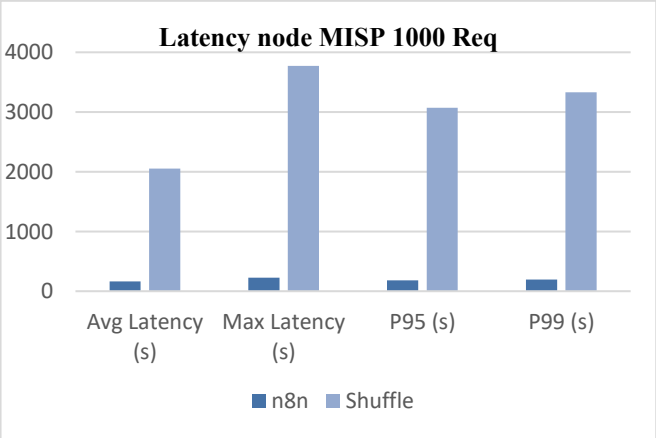
Secara keseluruhan, seluruh *node* utama pada n8n menunjukkan *latency* rata-rata yang lebih rendah dibandingkan Shuffle pada skenario 100 *request*. Temuan ini mengindikasikan bahwa n8n mampu mengeksekusi tahapan *workflow* SOC secara lebih baik pada kondisi beban ringan.

TABEL XI
HASIL LATENCY PER NODE PADA SKENARIO 1000 REQUEST

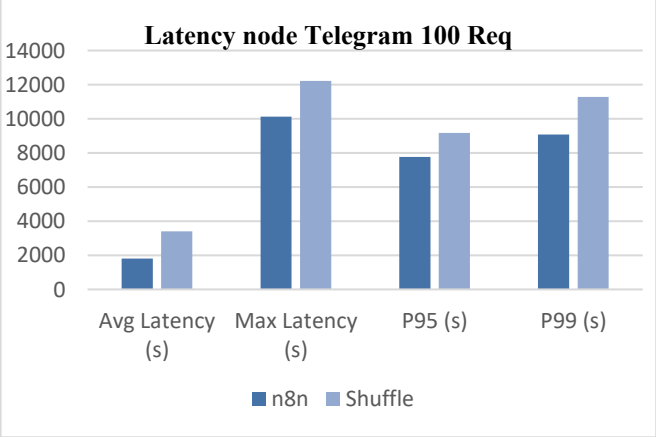
Platform	Node	Avg Latency (ms)	Max Latency (ms)	P95 (ms)	P99 (ms)
n8n	Active Response	103,39	329	119	158
n8n	MISP Check IP	161,903	226	182	193
n8n	Telegram Notification	1806,894	10122	7770	9074
Shuffle	Active Response	1291,449	2535	2011	2200
Shuffle	MISP Check IP	2051,281	3771	3073	3332
Shuffle	Telegram Notification	3407,143	12212	9177	11288



Gambar 17 Latency node active response 1000 Req



Gambar 18 Latency node MISP 1000 Req



Gambar 19 Latency node Telegram 1000 Req

Berdasarkan Tabel XI dan Gambar 17 sampai Gambar 19, pola yang sama juga terlihat pada skenario 1000 *request*. *Node* Telegram *Notification* tetap menjadi tahapan dengan *latency* tertinggi pada kedua platform. Pada n8n, rata-rata *latency* Telegram *Notification* sebesar 1.806,89 ms, sedangkan pada Shuffle mencapai 3.407,14 ms. Hasil ini menunjukkan bahwa proses pengiriman notifikasi tetap menjadi faktor dominan yang memengaruhi total *latency workflow* ketika jumlah *request* meningkat.

Pada *node Active Response*, n8n menghasilkan rata-rata *latency* sebesar 103,39 ms, sedangkan Shuffle sebesar 1.291,45 ms. Sementara itu, *node* MISP *Check* IP pada n8n memiliki rata-rata *latency* sebesar 161,90 ms, sedangkan Shuffle mencapai 2.051,28 ms. Perbedaan tersebut menunjukkan bahwa proses respons otomatis dan integrasi *Threat Intelligence* pada Shuffle secara konsisten membutuhkan waktu eksekusi yang lebih tinggi dibandingkan n8n.

Nilai p95 dan p99 juga memperlihatkan perbedaan karakteristik distribusi *latency* antar platform. Pada *node* Telegram *Notification*, n8n menghasilkan nilai p95 sebesar 7.770 ms dan p99 sebesar 9.074 ms, sedangkan Shuffle menghasilkan nilai p95 sebesar 9.177 ms dan p99 sebesar

e-ISSN: 2548-6861

11.288 ms. Hasil ini menunjukkan bahwa pada kondisi beban tinggi, sebagian besar *request* pada n8n tetap dapat diselesaikan dalam waktu yang lebih rendah dibandingkan Shuffle.

Jika dibandingkan dengan skenario 100 request, peningkatan jumlah *request* menjadi 1000 request tidak menyebabkan perubahan yang signifikan pada rata-rata *latency Active Response* dan MISP *Check* IP di kedua platform. Namun demikian, terjadi peningkatan *latency* pada node Telegram Notification, terutama pada Shuffle. Hal ini menunjukkan bahwa integrasi dengan layanan eksternal memiliki pengaruh yang lebih besar terhadap performa *workflow* dibandingkan proses internal workflow itu sendiri [9].

Secara keseluruhan, hasil pengujian menunjukkan bahwa n8n mempertahankan performa yang lebih konsisten pada seluruh *node* utama dibandingkan Shuffle. Perbedaan performa terbesar terlihat pada *node Active Response* dan MISP *Check* IP, di mana *latency* rata-rata Shuffle berada pada kisaran dua belas kali lebih tinggi dibandingkan n8n.

Berdasarkan hasil pengujian pada kedua skenario, *node* Telegram *Notification* merupakan komponen yang memberikan kontribusi terbesar terhadap total *latency workflow* pada kedua platform. Temuan ini menunjukkan bahwa komunikasi dengan layanan eksternal melalui API Telegram menjadi faktor dominan dalam menentukan waktu penyelesaian *workflow* SOC secara keseluruhan [9].

Selain itu, perbedaan performa antara n8n dan Shuffle tidak hanya terlihat pada *node* Telegram *Notification*, tetapi juga pada *node Active Response* dan MISP *Check* IP. Pada kedua node tersebut, n8n secara konsisten menghasilkan *latency* yang jauh lebih rendah dibandingkan Shuffle. Hal ini mengindikasikan bahwa mekanisme eksekusi *workflow* pada n8n lebih baik dalam menangani proses orkestrasi dan integrasi antarlayanan.

Dengan demikian, hasil *latency* per *node* mendukung temuan pada pengujian total *latency workflow* yang menunjukkan bahwa n8n mampu memberikan waktu respons yang lebih rendah dan lebih stabil dibandingkan Shuffle. Bagi implementasi SOC yang membutuhkan respons cepat terhadap *alert* keamanan, karakteristik ini menjadi faktor penting karena dapat mempercepat proses deteksi, validasi, dan respons terhadap insiden keamanan yang terjadi.

E. Pembahasan Perbandingan n8n dan Shuffle

Berdasarkan seluruh hasil pengujian yang telah dilakukan, n8n menunjukkan performa yang lebih baik dibandingkan Shuffle pada sebagian besar metrik yang dievaluasi, yaitu CPU *usage*, RAM *usage*, total *latency workflow*, *latency* per *node*, *throughput*, serta *tail latency* (p95 dan p99). Hasil ini menunjukkan bahwa terdapat perbedaan karakteristik performa antara kedua platform ketika digunakan untuk menjalankan *workflow* SOC yang sama.

Dari sisi penggunaan sumber daya, n8n menunjukkan nilai CPU *usage* dan RAM *usage* yang lebih rendah dibandingkan Shuffle pada kedua skenario pengujian. Pada skenario 100 *request*, rata-rata penggunaan CPU n8n sebesar 2,30%, sedangkan Shuffle mencapai 28,24%. Pada skenario 1000 *request*, penggunaan CPU n8n sebesar 1,96% dan Shuffle sebesar 16,21%. Pola yang sama juga terlihat pada penggunaan RAM, di mana n8n menggunakan RAM rata-rata sebesar 4.920,37 MB pada skenario 100 *request* dan 4.968,06 MB pada skenario 1000 *request*, sedangkan Shuffle menggunakan RAM rata-rata sebesar 8.477,44 MB dan 9.118,83 MB.

Dari sisi waktu respons, n8n menghasilkan total *latency workflow* yang lebih rendah dibandingkan Shuffle pada kedua skenario pengujian. Pada skenario 100 *request*, rata-rata total *latency workflow* n8n sebesar 2,36 detik, sedangkan Shuffle sebesar 12,95 detik. Pada skenario 1000 *request*, rata-rata *latency* n8n sebesar 2,476 detik dan Shuffle sebesar 12,94 detik. Nilai *latency* maksimum, p95, dan p99 juga menunjukkan pola yang sama, di mana n8n secara konsisten menghasilkan nilai yang lebih rendah dibandingkan Shuffle [15].

Hasil pengujian *throughput* menunjukkan bahwa n8n mampu memproses request lebih banyak dibandingkan Shuffle pada lingkungan pengujian yang digunakan. Pada skenario 100 request, throughput n8n mencapai 0,4237 *request* per detik, sedangkan Shuffle sebesar 0,2667 *request* per detik. Pada skenario 1000 *request*, throughput n8n sebesar 0,4234 *request* per detik dan Shuffle sebesar 0,3420 *request* per detik. Nilai *throughput* n8n juga relatif konsisten pada kedua skenario pengujian.

Hasil pengujian *latency* per node menunjukkan bahwa Telegram *Notification* merupakan komponen yang memberikan kontribusi terbesar terhadap total *latency* workflow pada kedua platform. Namun demikian, perbedaan performa terbesar antara n8n dan Shuffle terlihat pada *node Active Response* dan MISP *Check* IP. Pada kedua node tersebut, *latency* rata-rata Shuffle secara konsisten lebih tinggi dibandingkan n8n baik pada skenario 100 *request* maupun 1000 *request*.

Temuan ini menunjukkan bahwa meskipun kedua platform mampu menjalankan *workflow* SOC yang terintegrasi dengan Wazuh, MISP, Wazuh *Active Response*, dan Telegram, karakteristik performa yang dihasilkan berbeda. Dalam lingkungan pengujian yang digunakan pada penelitian ini, n8n menghasilkan penggunaan CPU dan RAM yang lebih rendah, *latency* yang lebih rendah, serta *throughput* yang lebih tinggi dibandingkan Shuffle.

Hasil penelitian ini juga melengkapi penelitian sebelumnya oleh Okroshidze yang membahas pemanfaatan *workflow* automation untuk kebutuhan SOC pada organisasi dengan keterbatasan anggaran. Berbeda dengan penelitian tersebut yang lebih berfokus pada aspek implementasi dan dampak operasional, penelitian ini menunjukkan adanya perbedaan performa yang terukur ketika kedua platform menjalankan workflow SOC yang identik.

Secara keseluruhan, berdasarkan metrik yang diuji pada penelitian ini, n8n menunjukkan nilai yang lebih baik dibandingkan Shuffle pada penggunaan CPU, penggunaan RAM, total *latency workflow*, *latency* per *node*, *throughput*, p95, dan p99. Hasil tersebut menunjukkan bahwa n8n mampu memberikan waktu respons yang lebih rendah dan penggunaan sumber daya yang lebih kecil dibandingkan Shuffle pada lingkungan pengujian yang digunakan.

F. Ringkasan Hasil Perbandingan

Ringkasan perbandingan performa n8n dan Shuffle berdasarkan seluruh metrik utama ditampilkan pada Tabel XII dan Tabel XIII.

TABEL XII
RINGKASAN PERBANDINGAN PADA SKENARIO 100 REQUEST

Metrik	n8n 100 Request	Shuffle 100 Request	Interpretasi
CPU Usage (%)	2,30	28,24	n8n lebih rendah
RAM Usage (MB)	4.920,37	8.477,44	n8n lebih rendah
Avg Latency workflow(s)	2,36	12,95	n8n lebih rendah
Max Latency workflow (s)	10	17	n8n lebih rendah
P95 Latency workflow(s)	7	16	n8n lebih rendah
P99 Latency workflow(s)	10	17	n8n lebih rendah
Throughput (req/s)	0,423728814	0,266666667	n8n lebih tinggi
Throughput (req/m)	25,42372881	16	n8n lebih tinggi

TABEL XIII
RINGKASAN PERBANDINGAN PADA SKENARIO 1000 REQUEST

Metrik	n8n 1000 Request	Shuffle 1000 Request	Interpretasi
CPU Usage (%)	1,96	16,21	n8n lebih rendah
RAM Usage (MB)	4.968,06	9.118,83	n8n lebih rendah
Avg Latency workflow(s)	2,476	12,94	n8n lebih rendah
Max Latency workflow (s)	11	24	n8n lebih rendah
P95 Latency workflow(s)	8	19	n8n lebih rendah
P99 Latency workflow(s)	10	22	n8n lebih rendah
Throughput (req/s)	0,423370025	0,341997264	n8n lebih tinggi
Throughput (req/m)	25,40220152	20,51983584	n8n lebih tinggi

Berdasarkan Tabel XII dan Tabel XIII, n8n menunjukkan hasil yang lebih baik dibandingkan Shuffle pada seluruh metrik utama yang diuji. Pada kedua skenario pengujian, n8n menghasilkan nilai CPU *usage*, RAM *usage*, *average latency workflow*, *maximum latency workflow*, p95 *latency*, dan p99 *latency* yang lebih rendah dibandingkan Shuffle. Selain itu, n8n juga menghasilkan *throughput* yang lebih tinggi pada skenario 100 *request* maupun 1000 *request*. Hasil tersebut menunjukkan bahwa pola performa yang diperoleh relatif konsisten meskipun jumlah request yang diproses meningkat.

IV. KESIMPULAN

Penelitian ini telah berhasil mengimplementasikan dan mengevaluasi performa workflow *Security Operations Center* (SOC) secara *real time* pada platform *open source* n8n dan Shuffle. Kedua platform terbukti mampu mengorkestrasikan alur respons insiden yang terintegrasi penuh dengan SIEM (Wazuh), *Threat Intelligence* (MISP), mekanisme respons otomatis (Wazuh *Active Response*), serta media notifikasi (Telegram).

Dari sisi efisiensi sumber daya komputasi, n8n menunjukkan keunggulan yang sangat signifikan dibandingkan Shuffle. Pada skenario beban tinggi (1.000 request), n8n mampu menekan penggunaan rata-rata CPU hingga sekitar 88% lebih rendah (1,96% berbanding 16,21%) dan konsumsi RAM hampir 45% lebih hemat dibandingkan Shuffle. Kestabilan n8n juga tervalidasi dari minimnya lonjakan konsumsi memori saat terjadi eskalasi jumlah request, menjadikannya opsi yang jauh lebih ideal untuk lingkungan SME yang umumnya memiliki keterbatasan infrastruktur.

Evaluasi terhadap waktu respons (*latency*) dan *throughput* semakin mempertegas keunggulan performa n8n. Secara keseluruhan, n8n mampu memproses workflow rata-rata lima

e-ISSN: 2548-6861

kali lebih cepat daripada Shuffle (2,476 detik berbanding 12,94 detik pada beban tinggi), dengan tingkat latensi maksimum dan tail latency (p95 serta p99) yang jauh lebih stabil. Analisis mendalam pada *latency* per *node* mengidentifikasi bahwa komunikasi dengan layanan API eksternal, yakni Telegram, merupakan penyumbang waktu tunda (*bottleneck*) terbesar di kedua platform. Meskipun demikian, arsitektur internal n8n terbukti mampu mengeksekusi integrasi kunci seperti automasi *Active Response* dan pengecekan MISP dengan waktu pemrosesan yang jauh lebih rendah dibandingkan Shuffle.

Secara komprehensif, dapat disimpulkan bahwa n8n merupakan platform automasi yang lebih responsif, stabil, dan hemat sumber daya untuk memenuhi kebutuhan operasional SOC pada skala pembiayaan SME. Untuk penelitian selanjutnya, disarankan untuk menguji performa kedua platform menggunakan metrik *concurrency* yang lebih esktrem, memvariasikan integrasi perangkat keamanan yang lebih kompleks, serta mengeksplorasi analisis *backend* arsitektur dari masing-masing platform yang menjadi pemicu utama terjadinya disparitas performa tersebut.

Penelitian selanjutnya dapat mengembangkan pengujian dengan variasi jumlah *request* yang lebih besar, pengujian *concurrency* yang berbeda, penggunaan layanan *threat intelligence* atau media notifikasi yang berbeda, serta implementasi pada spesifikasi *server* yang beragam untuk memperoleh gambaran performa yang lebih komprehensif pada berbagai kondisi operasional. Untuk meningkatkan validitas data, pengujian mendatang perlu menerapkan beberapa skenario percobaan berulang yang hasilnya dirata-rata guna meminimalisir dampak anomali sistem. Selain itu, evaluasi komparatif perlu dilengkapi dengan uji statistika signifikansi untuk membuktikan secara ilmiah apakah perbedaan metrik performa antara n8n dan Shuffle benar-benar signifikan secara statistik.

DAFTAR PUSTAKA

[1] M. Vielberth, F. Böhm, I. Fichtinger and G. Pernul, "Security Operations Center: A Systematic Study and Open Challenges," *IEEE Access,* vol. 8, pp. 227756-227779, 2020.

[2] P. Cichonski, T. Millar, T. Grance and K. Scarfone, "Computer Security Incident Handling Guide," *NIST Special Publication (SP) 800-61 Revision 2,* 2012.

[3] P. Kearney, M. Abdelsamea, X. Schmoor, F. Shah and I. Vickers, "Combating Alert Fatigue in the Security Operations Centre," *SSRN Electronic Journal,* 2023.

[4] W. U. Hassan, S. Guo, D. Li, Z. Chen, K. Jee, Z. Li and A. Bates, "NoDoze: Combatting Threat Alert Fatigue with Automated Provenance Triage," *Network and Distributed Systems Security (NDSS) Symposium,* 2019.

[5] C. Islam, M. B. Ali and S. Nepal, "A Multi-Vocal Review of Security Orchestration," *ACM Computing Surveys,* vol. 52, no. 2, pp. 1-45, 2019.

[6] I. R. Kurnia, Z. A. Brata, G. A. Nelistiani, S. Heo, H. Kim and H. Kim, "Toward Robust Security Orchestration and Automated Response in Security Operations Centers with a Hyper-Automation Approach Using Agentic Artificial Intelligence," *MDPI,* 2025.

[7] K. Knerler, I. Parker and C. Zimmerman, 11 Strategies of a World-Class Cybersecurity Operations Center, The MITRE Corporation, 2022.

[8] J. Manzoor, A. Waleed, A. F. Jamali and A. Masood, "Cybersecurity on a budget: Evaluating security and performance of open-source SIEM solutions for SMEs," *PLoS ONE,* 2024.

[9] R. T. Fielding and R. N. Taylor, "Principled Design of the ModernWeb Architecture," *ACM Transactions on Internet Technology,* vol. 2, no. 2, pp. 115-150, 2002.

[10] M. Landauer, F. Skopik, M. Wurzenberger and A. Rauber, "System log clustering approaches for cyber security applications: A survey," *Computer & Security 92,* 2020.

[11] G. Okroshidze, "Comparison of SOC Workflow Automation Options for SMEs and Practical Impact Analysis," 2024.

[12] J. Scheuner and P. Leitner, "Function-as-a-Service Performance Evaluation: A Multivocal Literature Review," *Journal of Systems and Software (JSS),* vol. 170, 2020.

[13] M. Kerrisk, "sar(1) — Linux manual page," [Online]. Available: https://man7.org/linux/man-pages/man1/sar.1.html.

[14] TIPHON, "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); General aspects of Quality of Service (QoS)," *ETSI TR 101 329 V2.1.1,* 1999.

[15] J. Dean and L. A. Barroso, "The Tail at Scale," *Communications of the ACM,* vol. 56, pp. 74-80, 2013.