



**OPTIMALISASI PROSES KEAMANAN PROAKTIF
MELALUI HUB MANAJEMEN MULTI-WAZUH
TERINTEGRASI *THREAT INTELLIGENCE* DAN
NOTIFIKASI *REAL-TIME***

SKRIPSI

**CHRISTIAN NATANIEL YOSUA PURBA
NIM 2207421043**

**PROGRAM STUDI TEKNIK MULTIMEDIA DAN
JARINGAN
JURUSAN TEKNIK INFORMATIKA DAN
KOMPUTER
POLITEKNIK NEGERI JAKARTA
TAHUN 2026**



**OPTIMALISASI PROSES KEAMANAN PROAKTIF
MELALUI HUB MANAJEMEN MULTI-WAZUH
TERINTEGRASI *THREAT INTELLIGENCE* DAN
NOTIFIKASI *REAL-TIME***

SKRIPSI

**Dibuat untuk Melengkapi Syarat-Syarat yang Diperlukan untuk
Memperoleh Diploma Empat Politeknik**

**CHRISTIAN NATANIEL YOSUA PURBA
NIM 2207421043**

**PROGRAM STUDI TEKNIK MULTIMEDIA DAN
JARINGAN
JURUSAN TEKNIK INFORMATIKA DAN
KOMPUTER
POLITEKNIK NEGERI JAKARTA
TAHUN 2026**



© Hak Cipta milik Politeknik Negeri Jakarta

SURAT PERNYATAAN BEBAS PLAGIARISME

yang bertanda tangan di bawah ini:

: Christian Nataniel Yosua Purba

: 2207421043

Jurusan/Program Studi: T.Informatika dan Komputer / T. Multimedia dan Jaringan

Judul skripsi : OPTIMALISASI PROSES KEAMANAN PROAKTIF
MELALUI HUB MANAJEMEN MULTI-WAZUH TERINTEGRASI THREAT
INTELLIGENCE DAN NOTIFIKASI REAL-TIME

Menyatakan dengan sebenarnya bahwa skripsi ini benar-benar merupakan hasil karya saya sendiri, bebas dari peniruan terhadap karya dari orang lain. Kutipan pendapat dan tulisan orang lain ditunjuk sesuai dengan cara-cara penulisan karya ilmiah yang berlaku.

Apabila di kemudian hari terbukti atau dapat dibuktikan bahwa dalam skripsi ini terkandung ciri-ciri plagiat dan bentuk-bentuk peniruan lain yang dianggap melanggar peraturan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Depok, 8 Juni, 2026

Yang membuat pernyataan


10000
METEOR
TEMPER
8666 CANX335253795

(Christian Nataniel Yosua Purba)

NIM. 2207421043

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber ;
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



LEMBAR PENGESAHAN

Skripsi diajukan oleh :
Nama : Christian Nataniel Yosua Purba
NIM : 2207421043
Program Studi : TMJ
Judul Skripsi : OPTIMALISASI PROSES KEAMANAN PROAKTIF
MELALUI HUB MANAJEMEN MULTI-WAZUH TERINTEGRASI *THREAT*
INTELLIGENCE DAN NOTIFIKASI *REAL-TIME*

Telah diuji oleh tim penguji dalam Sidang Skripsi pada hari Senin Tanggal 22
Bulan Juni Tahun 2026 dan dinyatakan LULUS.

Disahkan oleh:

Pembimbing I : Ayu Rosyida Zain, S.ST., MT

Penguji I : Dr. Defiana Arnaldy, S.Tp., M.Si.

Penguji II : Iik Muhamad Malik Matin, S.Kom., M.T.

Penguji III : Andika Riyadi Jasril, S.Pd., M.Pd.T

Mengetahui:

Jurusan Teknik Informatika dan Komputer

Ketua



Dr Anita Hidayati S Kom M Kom

NIP. 197908032003122003

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



KATA PENGANTAR

Puji syukur kepada Tuhan Yesus Kristus karena hanya oleh anugerah dan kasih sayang-Nya penulis dapat melaksanakan perkuliahan dengan baik dan sampai kepada penyelesaian penulisan Skripsi yang berjudul "Optimalisasi Proses Keamanan Proaktif melalui Hub Manajemen Multi-Wazuh Terintegrasi *Threat Intelligence* dan Notifikasi *Real-Time*". Skripsi ini disusun sebagai salah satu syarat untuk meraih gelar Sarjana Terapan pada Program Studi Teknik Multimedia dan Jaringan, Jurusan Teknik Informatika dan Komputer, Politeknik Negeri Jakarta.

Penulis menyadari bahwa perjalanan panjang dalam menyusun penelitian ini bukanlah jalan yang mudah untuk dilalui sendiri. Terdapat banyak doa, bimbingan, pengorbanan, dan dukungan tanpa henti dari berbagai pihak yang memungkinkan penulis untuk tiba di titik ini. Oleh karena itu, dengan kerendahan hati penulis ingin menyampaikan rasa terima kasih yang mendalam kepada:

1. Almh. Ibunda tercinta, sosok luar biasa yang telah membentuk karakter penulis, dan yang kasih sayangnya terus mengalir menjadi kekuatan yang mengiringi langkah penulis hingga akhir hayat.
2. Keluarga besar penulis, secara khusus Opung, Tante, dan Tulang serta Bapak yang senantiasa mendoakan dan memberikan dukungan yang luar biasa, sehingga memungkinkan penulis untuk memiliki kesempatan menempuh pendidikan hingga di bangku perkuliahan ini.
3. Ibu Ayu Rosyida Zain, S.ST., M.T., selaku dosen pembimbing yang telah mendedikasikan waktu, tenaga, serta kesabarannya untuk memberikan arahan dan masukan yang sangat berharga selama proses penyusunan skripsi ini.
4. Rekan seperjuangan kuliah, kerja dan skripsi, Pratama dan Abian. Terima kasih untuk segala suka dan duka, serta momen ramai maupun sepi yang kita lewati bersama demi menyelesaikan sistem dan penelitian ini hingga tuntas.

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

5. Rekan-rekan PBL Group lainnya, yaitu Azzuri, Ayu, Caca, Izza, dan Rei karena tanpa kehadiran dan kebersamaan kalian, perjalanan perkuliahan penulis tidak akan seberwarna dan bermakna ini.
6. Teman-teman seiman dalam Connect Group, POSA PNJ, dan PBI Kartini, yang tidak dapat disebutkan satu per satu, atas setiap doa, penguatan rohani, dan kehangatan persaudaraan yang terus menguatkan penulis.
7. Seluruh teman-teman seperjuangan Teknik Multimedia dan Jaringan angkatan 2022 (TMJ 22), atas kebersamaan dan kenangan indah selama menempuh pendidikan di Politeknik Negeri Jakarta.

Penulis menyadari bahwa Skripsi ini masih memiliki kekurangan dan jauh dari kata sempurna. Oleh karena itu, kritik dan saran yang membangun sangat penulis harapkan demi penyempurnaan ke depannya. Akhir kata, semoga penelitian ini dapat memberikan manfaat nyata bagi pengembangan ilmu pengetahuan, khususnya di bidang keamanan siber.

Depok, 8 Juni 2026

Christian Nataniel Yosua Purba

**POLITEKNIK
NEGERI
JAKARTA**



© Hak Cipta milik Politeknik Negeri Jakarta

SURAT PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Politeknik Negeri Jakarta, saya bertanda tangan dibawah

ini

Nama

: Christian Nataniel Yosua Purba

NI

: 2207421043

Jurusan/Program Studi: T. Informatika dan Komputer / T. Multimedia dan Jaringan

Demikian pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada

Politeknik Negeri Jakarta Hak Bebas Royalti Non-Eksklusif atas karya ilmiah

saya yang berjudul :

**OPTIMALISASI PROSES KEAMANAN PROAKTIF MELALUI HUB
MANAJEMEN MULTI-WAZUH TERINTEGRASI THREAT
INTELLIGENCE DAN NOTIFIKASI REAL-TIME**

Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non-Eksklusif ini Politeknik Negeri Jakarta berhak menyimpan, mengalihmediakan/formatkan, mengelola dalam bentuk pangkalan data (database), merawat, dan mempublikasikan skripsi saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Depok, 08 Juni 2026

Yang menyatakan

Christian Nataniel Yosua Purba



- Hak Cipta :**
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
 2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



OPTIMALISASI PROSES KEAMANAN PROAKTIF MELALUI HUB MANAJEMEN MULTI-WAZUH TERINTEGRASI *THREAT INTELLIGENCE* DAN NOTIFIKASI *REAL-TIME*

ABSTRAK

Security Operations Center (SOC) menghadapi tantangan penyempitan breakout time akibat fragmentasi infrastruktur keamanan dan tingginya beban kognitif analis dari perpindahan konteks (context-switching) saat investigasi manual. Penelitian ini bertujuan untuk mengoptimalkan proses keamanan proaktif melalui rancang bangun Hub Manajemen Terpusat berbasis arsitektur cache-database (PostgreSQL dan Redis). Sistem ini mengintegrasikan komponen multi-Wazuh sebagai Single Source of Truth (SSOT), layanan Threat Intelligence (AlienVault OTX) secara asinkron, dan modul background scheduler (Sentinel) untuk notifikasi real-time via Telegram. Metode penelitian menggunakan System Engineering dengan pendekatan eksperimental kuantitatif. Pengujian fungsional Black Box dan non-fungsional memastikan sistem tangguh dalam menangani anomali jaringan dan mengamankan kredensial dengan enkripsi AES. Hasil pengujian efisiensi operasional membuktikan bahwa sistem usulan mampu mentransformasi pemantauan pasif menjadi proaktif. Evaluasi komparatif menunjukkan bahwa Hub Manajemen ini berhasil memangkas rata-rata waktu operasional sebesar 58,33% dan mereduksi beban interaksi analis sebesar 52,0%. Kesimpulannya, sistem ini sukses mengeliminasi redundansi kerja manual dan mengakselerasi respons investigasi SOC secara signifikan.

Kata kunci: AlienVault, cache-database, notifikasi real-time, Redis, SOAR, threat intelligence, Wazuh.

POLITEKNIK
NEGERI
JAKARTA



DAFTAR ISI

LEMBAR PENGESAHAN.....	iii
KATA PENGANTAR.....	iv
ABSTRAK.....	vii
DAFTAR ISI.....	viii
DAFTAR GAMBAR.....	ix
DAFTAR TABEL.....	xiv
BAB I	
PENDAHULUAN.....	1
1.1 Latar Belakang Masalah.....	1
1.2 Perumusan Masalah.....	4
1.3 Batasan Masalah.....	4
1.4 Tujuan dan Manfaat.....	5
1.5 Sistematika Penulisan.....	6
BAB II	
TINJAUAN PUSTAKA.....	7
2.1 Security Operations Center (SOC) Proaktif.....	7
2.2 Efisiensi dan Optimalisasi Alur Kerja SOC.....	7
2.3 Security Information and Event Management (SIEM).....	8
2.4 Integrasi SOAR, SIEM dan <i>Firewall</i>	8
2.5 Wazuh.....	9
2.6 Wazuh <i>Agent</i> , Wazuh <i>Manager</i> , dan Wazuh <i>Indexer</i>	9
2.7 Alienvault OTX sebagai <i>Threat Intelligence</i>	11
2.8 Arsitektur OLTP <i>Cache-Database</i>	12
2.9 PostgreSQL.....	13
2.10 Redis.....	13
2.11 Telegram Bot API.....	14
2.12 Pengujian <i>Black Box Testing</i>	14
2.13 Transformasi <i>As-Is</i> dan <i>To-Be</i> untuk Operasional SOC.....	15
2.14 Standar Parameter Waktu Respon dan Kebutuhan Latensi <i>Real-Time</i>	15
2.15 Diagram Sekuensial dengan Fragmen Kombinasi.....	16
2.16 Penelitian Terkait.....	20
BAB III	
METODE PENELITIAN.....	23
3.1 Rancangan Penelitian.....	23
3.2 Tahapan Penelitian.....	23
3.3 Objek Penelitian.....	25
BAB IV	
HASIL DAN PEMBAHASAN.....	27
4.1 Analisis Kebutuhan.....	27



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

4.1.1 Kebutuhan Fungsional (Functional Requirements).....	27
4.1.2 Analisis Kebutuhan Non-Fungsional.....	29
4.1.3 Analisis Kebutuhan Perangkat Keras (<i>Hardware</i>).....	31
4.1.4 Analisis Kebutuhan Perangkat Lunak (<i>Software</i>).....	32
4.2 Perancangan Sistem.....	34
4.2.1 Arsitektur Perangkat Lunak dan Aliran Data.....	34
4.2.2 Rancangan Optimalisasi Alur Kerja Operasional (<i>As-Is</i> dan <i>To-Be</i>)	36
4.2.3 Perancangan Penyimpanan Data (<i>Cache-Database Architecture</i>)....	57
4.3 Implementasi Sistem.....	63
4.3.1 Implementasi Lingkungan Operasional (Orkestrasi Kontainer).....	63
4.3.2 Implementasi Hub Manajemen Konfigurasi Multi-Wazuh.....	68
4.3.3 Implementasi Hub Agregasi Multi-Agen Wazuh.....	85
4.3.4 Implementasi Pengayaan Intelijen Ancaman (<i>Threat Intelligence</i>)..	95
4.3.5 Implementasi Pemantauan Proaktif dan Notifikasi <i>Real-Time</i> (Telegram).....	108
4.3.6 Implementasi Hak Akses Berbasis Peran (<i>Role-Based Access Control</i>)...	124
4.4 Pengujian.....	126
4.4.1 Deskripsi Pengujian.....	126
4.4.2 Prosedur Pengujian.....	127
A. Prosedur Pengujian Fungsional (<i>Black Box Testing</i>).....	128
B. Prosedur Pengujian Non-Fungsional.....	136
C. Prosedur Pengujian Efisiensi Operasional.....	140
4.4.3 Data Hasil Pengujian.....	149
A. Data Hasil Pengujian Fungsional (<i>Black Box Testing</i>).....	150
B. Data Hasil Pengujian Non-Fungsional.....	155
C. Data Hasil Pengujian Efisiensi Operasional.....	160
4.4.4 Analisis Data / Evaluasi Pengujian.....	162
A. Analisis Pengujian Fungsional (<i>Black Box Testing</i>).....	163
B. Analisis Pengujian Non-Fungsional.....	164
C. Analisis dan Evaluasi Efisiensi Operasional.....	168
BAB V	
PENUTUP.....	176
5.1. Simpulan.....	176
5.2. Saran.....	177
DAFTAR RIWAYAT HIDUP PENULIS.....	178
DAFTAR PUSTAKA.....	179
LAMPIRAN.....	180



- Hak Cipta :**
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
 2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

DAFTAR GAMBAR

Gambar 2.1 Ilustrasi Data Multi-Sumber pada Data <i>Aggregation Module</i>	8
Gambar 2.2 Ilustrasi berbagai Kapabilitas SOAR.....	9
Gambar 2.3 Topologi Arsitektural Komponen Wazuh.....	10
Gambar 2.4 Fungsi-Fungsi Komponen Wazuh.....	10
Gambar 2.5 Struktur Model Data Alienvault OTX.....	12
Gambar 2.6 Diagram Mekanisme Implementasi <i>Cache-Database</i>	13
Gambar 2.8 Contoh Fragment Gabungan <i>Break</i> dan <i>Loop</i> dalam Satu <i>Sequential</i> ..	17
Gambar 2.9 Contoh Diagram Sekuensial dengan Alternatives.....	18
Gambar 2.10 Contoh Diagram Sekuensial dengan Operator Parallel.....	19
Gambar 2.11 Contoh Diagram Sekuensial dengan Operator <i>Loop</i>	19
Gambar 3.1 Diagram Tahapan Penelitian.....	23
Gambar 3.2 Diagram Arsitektur Objek Penelitian.....	26
Gambar 4.1 Diagram Arsitektur Hub Manajemen Keamanan Terpusat.....	34
Gambar 4.2 Diagram Alur Komparasi Aktivitas Inventarisasi Komponen Baru Metode Manual (Existing) dan Metode Usulan (<i>Proposed</i>).....	39
Gambar 4.3 Diagram Alur Komparasi Aktivitas Pemeliharaan Kredensial (Perubahan <i>Password</i>) Metode Manual (Kiri) dan Metode Usulan (Kanan).....	41
Gambar 4.4 Diagram Alur Komparasi Aktivitas Pemantauan Kesehatan Multi-Komponen Metode Manual (Kiri) dan Metode Usulan (Kanan).....	43
Gambar 4.5 Diagram Alur Komparasi Aktivitas Inventarisasi Daftar Agen (<i>Agent List</i>) Metode Manual (Kiri) dan Metode Usulan (Kanan).....	45
Gambar 4.6 Diagram Alur Komparasi Aktivitas Investigasi On-Demand Metode Manual (Kiri) dan Metode Usulan (Kanan).....	47
Gambar 4.7 Diagram Alur Komparasi Aktivitas Investigasi Alert (<i>Click-to-Investigate</i>) Metode Manual (Kiri) dan Metode Usulan (Kanan).....	49
Gambar 4.8 Diagram Alur Komparasi Aktivitas Konfigurasi API <i>Threat Intelligence</i> Metode Manual (Kiri) dan Metode Usulan (Kanan).....	51
Gambar 4.9 Diagram Alur Komparasi Aktivitas Konfigurasi Notifikasi Metode Manual (Kiri) dan Metode Usulan (Kanan).....	53
Gambar 4.10 Diagram Alur Komparasi Aktivitas Uji Coba Integrasi (<i>Connectivity Test</i>) Metode Manual (Kiri) dan Metode Usulan (Kanan).....	55
Gambar 4.11 Diagram Alur Komparasi Aktivitas Pemantauan Status Komponen Metode Manual (Kiri) dan Metode Usulan (Kanan).....	57
Gambar 4.12 Diagram Ilustrasi Orkestrasi Kontainer dalam Docker.....	63
Gambar 4.13 Potongan Konfigurasi Bagian PostgreSQL docker-compose.yaml.....	65
Gambar 4.14 Potongan Konfigurasi Bagian Redis docker-compose.yaml.....	66
Gambar 4.15 Potongan Konfigurasi Bagian Hub docker-compose.yaml.....	67
Gambar 4.16 Potongan Konfigurasi <i>Background Scheduler</i> docker-compose.yaml.....	67



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Gambar 4.17 Konfigurasi Bagian Volume <i>Global</i> docker-compose.yaml.....	68
Gambar 4.18 Diagram Alur Sekuensial Penambahan Komponen Berhasil.....	70
Gambar 4.19 Diagram Alur Sekuensial Penambahan Komponen Gagal.....	71
Gambar 4.20 Antarmuka Manajemen Komponen Infrastruktur (Wazuh <i>Indexer</i>)...	72
Gambar 4.21 Antarmuka Manajemen Komponen Infrastruktur (Wazuh <i>Manager</i>)	72
Gambar 4.22 Raw Response <i>Unauthorized</i> untuk Wazuh <i>Manager</i> dan <i>Indexer</i>	73
Gambar 4.23 Raw Response Generasi <i>Token</i> JWT pada Wazuh <i>Manager</i> via CLI.	73
Gambar 4.24 Raw Response Data Log dari Wazuh <i>Indexer</i> via CLI.....	74
Gambar 4.25 Potongan Kode Konektor Wazuh <i>Indexer</i> (HTTP Basic <i>Authentication</i>).....	75
Gambar 4.26 Potongan Konektor Wazuh <i>Manager</i> (JWT & <i>Caching</i>).....	77
Gambar 4.27 Potongan Kode Model Data <i>WazuhManager</i>	78
Gambar 4.28 Tampilan Table <i>wazuh_indexers</i>	78
Gambar 4.29 Tampilan Table <i>wazuh_managers</i>	78
Gambar 4.30 Potongan Kode <i>Controller</i> Hub Multi-Wazuh.....	79
Gambar 4.31 Potongan Kode Logika Edit dan <i>Delete</i> Hub Multi-Wazuh.....	81
Gambar 4.32 Tampilan Notifikasi <i>Toast</i> saat Mengedit Komponen.....	81
Gambar 4.33 Tampilan Notifikasi <i>Toast</i> saat Update Komponen Berhasil.....	82
Gambar 4.34 Tampilan <i>Modal</i> untuk Konfirmasi Penghapusan Komponen.....	82
Gambar 4.35 Tampilan <i>Toast</i> Notifikasi Saat Berhasil Menghapus Komponen..	82
Gambar 4.36 Potongan Kode Logika Frontend <i>Refresh</i>	83
Gambar 4.37 Tampilan Notifikasi <i>Toast</i> saat <i>Refresh</i> dari <i>Cache</i>	84
Gambar 4.38 Tampilan Notifikasi <i>Toast</i> saat <i>Refresh Live Status</i>	84
Gambar 4.39 Tampilan Status Indikator saat Komponen Tersambung.....	84
Gambar 4.40 Tampilan Status Indikator saat Komponen Terputus.....	85
Gambar 4.41 Proses Manual Penarikan Data Agen Wazuh via CLI (Sistem Lama <i>As-Is</i>).....	86
Gambar 4.42 Diagram Alur Sekuensial Sistem Penarikan Data Multi-Agen secara Paralel (Sistem <i>Proposed</i>).....	87
Gambar 4.43 Potongan Fungsi Kode Agregasi Paralel.....	89
Gambar 4.44 Potongan Kode Fungsi Pengayaan Data di dalam <i>Thread</i>	89
Gambar 4.45 <i>Field Manager</i> Name yang diambil dari enrichment.....	90
Gambar 4.46 Diagram Abstraksi Mekanisme <i>Caching</i> Operasional.....	91
Gambar 4.47 Potongan Kode Logika Redis <i>Caching</i>	92
Gambar 4.48 Tampilan UI Wazuh <i>Agents</i>	92
Gambar 4.49 Tampilan Pencarian dan Filtering <i>Agent</i>	93
Gambar 4.50 Potongan Kode JS Logika URL <i>State Management</i>	94
Gambar 4.51 Tautan Interaktif Agen menuju <i>Dashboard</i> Analisis.....	94



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Gambar 4.52 Panel Interaktif <i>Connected Agents</i> di Dashboard.....	95
Gambar 4.53 Antarmuka Web AlienVault OTX untuk Mencari Satu IP.....	96
Gambar 4.54 Antarmuka Tab Analysis di Web AlienVault OTX.....	96
Gambar 4.55 Antarmuka Tab Pulses di Web AlienVault OTX.....	97
Gambar 4.56 Respons Mentah (Raw JSON) dari API AlienVault OTX via Terminal.....	97
Gambar 4.57 Diagram Alur Sekuensial Investigasi Asinkron (<i>Threat Intelligence</i>).....	99
Gambar 4.58 Potongan Kode Fungsi Konektor dan Penarikan Data OTX.....	101
Gambar 4.59 Potongan Model Penyimpanan <i>Threat Intelligence</i>	103
Gambar 4.60 Bentuk <i>Ciphertext</i> dari <i>encrypted_api_key</i>	103
Gambar 4.61 Bagian Tab <i>Intelligence</i> di Dashboard.....	104
Gambar 4.62 Hasil <i>Threat Intelligence</i> di Modul untuk Pencarian IoC IPv4....	104
Gambar 4.63 Hasil <i>Threat Intelligence</i> di Modul untuk Pencarian IoC domain	105
Gambar 4.64 Tampilan Related Threat Pulses.....	106
Gambar 4.65 Komponen Interaktif JSON <i>Document Explorer</i>	107
Gambar 4.66 Tombol Lookup untuk <i>Shortcut Intelligence</i> pada Dashboard....	108
Gambar 4.67 Diagram Alur Sekuensial Pemantauan Latar Belakang (Sentinel)....	109
Gambar 4.68 Potongan Kode Fungsi Utama Pemantauan.....	111
Gambar 4.69 Potongan Kode Fungsi Evaluasi Status <i>UP DOWN</i> Komponen..	113
Gambar 4.70 Diagram Alur Sekuensial Pengiriman Pesan Asinkron (Pub/Sub)....	114
Gambar 4.71 Potongan Kode Konektor API Telegram.....	115
Gambar 4.72 Tampilan Pembuatan Bot Telegram dengan BotFather.....	116
Gambar 4.73 Tampilan Grup Telegram.....	117
Gambar 4.74 Tampilan URL untuk Mendapatkan Chat ID Grup Telegram.....	117
Gambar 4.75 Tampilan <i>Field Configuration</i> untuk Input Bot Token dan Chat IDs.	118
Gambar 4.76 Tampilan <i>Toast</i> Sukses dan Keseluruhan Page Telegram Integration	118
Gambar 4.77 Tampilan Notifikasi <i>Test Message</i>	119
Gambar 4.78 Potongan Kode Fungsi Pemformatan Teks Telegram.....	120
Gambar 4.79 Perintah CLI untuk Mensimulasikan Skenario B (Pemadaman Server Target).....	121
Gambar 4.80 Umpan Balik Peringatan Kritis dan Pemulihan pada Telegram...	122
Gambar 4.81 Peringatan Berkala (Reminder) pada Insiden Pemadaman Jangka Panjang.....	123
Gambar 4.82 Banner Peringatan Kegagalan Proses Pemantauan Latar Belakang...	123
Gambar 4.83 Halaman Login Sistem Hub Manajemen Terpusat.....	124



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengemukakan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Gambar 4.84 Tampilan Antarmuka Manajemen Pengguna Khusus Administrator.	125
Gambar 4.85 Adaptasi Dinamis Menu Navigasi Peran <i>Executor</i>	125
Gambar 4.86 Adaptasi Dinamis Menu Navigasi Peran <i>Viewer</i>	126
Gambar 4.87 Grafik Komparasi Waktu Muat Antarmuka.....	164
Gambar 4.88 Grafik Komparasi Waktu Eksekusi Asinkron dan Timeout.....	165
Gambar 4.89 Grafik Komparasi Waktu Latensi Notifikasi.....	166
Gambar 4.86 Grafik Komparasi Langkah per Aksi Operasional Eksisting & Usulan.....	171
Gambar 4.87 Grafik Komparasi detik per Operasional Eksisting & Usulan.....	172





DAFTAR TABEL

Tabel 2.1. Perbandingan Karakteristik Komponen Arsitektur Wazuh.....	11
Tabel 2.2. Perbandingan Penelitian Terkait.....	20
Tabel 4.1. Daftar Kebutuhan Perangkat Keras (<i>Hardware</i>).....	31
Tabel 4.2. Daftar Kebutuhan Perangkat Lunak (<i>Software</i>).....	32
Tabel 4.3. Perbandingan Alur Kerja Inventarisasi Komponen Baru.....	38
Tabel 4.4. Perbandingan Alur Kerja Pemeliharaan Kredensial.....	40
Tabel 4.5. Perbandingan Alur Kerja Pemantauan Kesehatan Multi-Komponen...	42
Tabel 4.6. Perbandingan Alur Kerja Inventarisasi Daftar Agen.....	44
Tabel 4.7. Perbandingan Alur Kerja Investigasi <i>On-Demand</i>	46
Tabel 4.8. Perbandingan Alur Kerja Investigasi <i>Alert (Click-to-Investigate)</i>	48
Tabel 4.9. Perbandingan Alur Kerja Konfigurasi API <i>Threat Intelligence</i>	50
Tabel 4.10. Perbandingan Alur Kerja Konfigurasi Notifikasi Telegram.....	52
Tabel 4.11. Perbandingan Alur Kerja Uji Coba Integrasi (<i>Connectivity Test</i>).....	54
Tabel 4.12. Perbandingan Alur Kerja Identifikasi Insiden <i>Down</i>	56
Tabel 4.13. Penjelasan Peran Redis dalam Sistem.....	58
Tabel 4.14. Penjelasan Spesifikasi <i>Key</i> Redis per Modul.....	60
Tabel 4.15. Penjelasan Spesifikasi <i>Key</i> Redis per Modul.....	62
Tabel 4.16. Spesifikasi Kontainer dan Alokasi Layanan.....	64
Tabel 4.17. Tabel Daftar Skenario dan Keadaan <i>State Machine</i>	112
Tabel 4.18. Skenario Pengujian Modul Sentralisasi Multi-Wazuh.....	128
Tabel 4.19. Skenario Pengujian Modul Integrasi <i>Threat Intelligence</i>	132
Tabel 4.20. Skenario Pengujian Modul Notifikasi Status Sistem.....	134
Tabel 4.21. Skenario Pengujian Waktu Respons (<i>Response Time</i>).....	137
Tabel 4.22. Skenario Pengujian Keamanan Data (<i>Security</i>).....	138
Tabel 4.23. Skenario Pengujian Keandalan (<i>Reliability & Fault Tolerance</i>).....	139
Tabel 4.24. Prosedur Pengujian Efisiensi Aksi 1 (Inventarisasi Komponen Baru)....	140
Tabel 4.25. Prosedur Pengujian Efisiensi Aksi 2 (Pemeliharaan Kredensial).....	141
Tabel 4.26. Prosedur Pengujian Efisiensi Aksi 3 (Pemantauan Kesehatan Multi-Komponen).....	142
Tabel 4.27. Prosedur Pengujian Efisiensi Aksi 4 (Inventarisasi Daftar Agen).....	143
Tabel 4.28. Prosedur Pengujian Efisiensi Aksi 5 (Investigasi <i>On-Demand</i>).....	144
Tabel 4.29. Prosedur Pengujian Efisiensi Aksi 6 (Investigasi <i>Alert</i>).....	145
Tabel 4.30. Prosedur Pengujian Efisiensi Aksi 7 (Konfigurasi API Intelijen).....	146
Tabel 4.31. Prosedur Pengujian Efisiensi Aksi 8 (Konfigurasi Notifikasi).....	146
Tabel 4.32. Prosedur Pengujian Efisiensi Aksi 9 (Uji Coba Integrasi).....	147
Tabel 4.33. Prosedur Pengujian Efisiensi Aksi 10 (Pemantauan Status).....	148
Tabel 4.34. Hasil Pengujian Modul Sentralisasi Multi-Wazuh.....	150

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengemukakan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengemukakan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Tabel 4.35 Hasil Pengujian Modul Sentralisasi <i>Threat Intelligence</i>	152
Tabel 4.36 Hasil Pengujian Modul Telegram Notifikasi Status Sistem.....	153
Tabel 4.37 Hasil Pengujian Waktu Muat Antarmuka (NF.1.1).....	155
Tabel 4.38 Hasil Pengujian Eksekusi Asinkron & <i>Timeout</i> (NF.1.2).....	156
Tabel 4.39 Hasil Pengujian Latensi Notifikasi (NF.1.3).....	157
Tabel 4.40 Hasil Pengujian Keamanan Data (NF.2).....	158
Tabel 4.41 Hasil Pengujian Keandalan (<i>Reliability</i>).....	159
Tabel 4.42 Rekapitulasi Data Waktu Penyelesaian (<i>Time on Task</i>).....	160
Tabel 4.43 Rekapitulasi Data Jumlah Langkah Interaksi (<i>Step Count</i>).....	161
Tabel 4.44 Tingkat Efisiensi (<i>Reduction Rate</i>) per Aksi Operasional.....	169





Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

BAB I PENDAHULUAN

1.1 Latar Belakang Masalah

Kemajuan digital yang memperluas konektivitas secara bersamaan juga memperlebar ruang serang siber. Sebagaimana ditegaskan oleh World Economic Forum (2025), "*As our digital footprints widen, so does the potential attack surface for nefarious actors.*" Di Indonesia, urgensi ini tercermin dalam lanskap keamanan siber yang mencatat total trafik anomali selama tahun 2024 mencapai 330.527.636 anomali (BSSN, 2024). Tren ini belum menunjukkan tanda-tanda penurunan di tahun 2025, di mana semester pertama saja telah mencatat lebih dari 133 juta serangan digital (AwanPintar.id, 2025).

Di tengah permasalahan ini, kecepatan respons insiden menjadi faktor yang sangat penting dalam meminimalkan dampak serangan. Hal ini berkaitan erat dengan konsep *breakout time*, yakni waktu yang dibutuhkan penyerang untuk bergerak secara lateral dari satu sistem yang terbobol ke bagian lain. Berdasarkan laporan CrowdStrike (2023), rata-rata *breakout time* berada pada angka 79 menit, kemudian menyusut drastis menjadi 62 menit pada 2024 dan terus menurun hingga tersisa 48 menit pada 2025 (CrowdStrike, 2024; 2025). Dari data ini jelas bahwa organisasi kini berpacu dalam hitungan menit untuk mencegah kerusakan sistemik.

Guna menghadapi jendela waktu yang semakin sempit tersebut, organisasi mengandalkan *Security Operations Center* (SOC) yang mengintegrasikan proses, teknologi, dan sumber daya manusia. Struktur SOC yang mumpuni sangat krusial dalam menciptakan *situational awareness*, memitigasi risiko yang terpapar (*exposed risks*), serta membantu organisasi memenuhi persyaratan regulasi yang berlaku (Vielberth et al., 2020). Namun, pendekatan reaktif tradisional yang hanya berfokus pada penanganan insiden setelah serangan terjadi terbukti tidak lagi memadai. Menurut Ofoegbu et al. (2024), terdapat kebutuhan mendesak untuk bergeser menuju strategi keamanan siber proaktif yang menekankan pada antisipasi dan mitigasi ancaman sebelum dampak serangan meluas ke operasional organisasi. Paradigma ini menuntut kemampuan deteksi proaktif yang berfokus



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

pada prediksi dan pencegahan melalui pemanfaatan teknik canggih serta integrasi teknologi yang berkelanjutan (Abiade, 2024).

Dengan banyaknya tugas yang ingin dicapai dalam waktu cepat, muncul sebuah pertanyaan kritis: apakah penambahan alat berbanding lurus dengan peningkatan kualitas pertahanan? Studi IBM dan Ponemon (2020) mengindikasikan bahwa perusahaan rata-rata menggunakan 45 perangkat keamanan yang berbeda. Ironisnya, studi yang sama mencatat organisasi yang dengan lebih dari 50 perangkat keamanan justru melaporkan tingkat kepercayaan yang lebih rendah dalam kemampuan deteksi dan respons mereka.

Kondisi banyaknya perangkat yang tidak terintegrasi ini memaksa analis melakukan korelasi informasi secara manual lintas sistem, sebuah proses yang memicu beban kognitif tinggi secara terus-menerus yang disebut dengan *context-switching*. Baruwat Chhetri et al. (2024) menegaskan bahwa volume peringatan yang membanjir tanpa korelasi dan konteks memadai menghambat proses pengambilan keputusan yang efektif saat terjadi serangan. Dampak dari fragmentasi ini divalidasi oleh penelitian Arif et al. (2025) yang menyimpulkan bahwa arsitektur yang terpecah-pecah berkontribusi langsung pada inefisiensi operasional dan eskalasi risiko keamanan. Penelitian tersebut secara eksplisit menyatakan urgensi pengembangan solusi yang terkonsolidasi dan berpusat pada pengguna (*user-centric*) guna mengatasi permasalahan ini.

Tantangan industri tersebut tercermin secara empiris di perusahaan yang dalam penelitian ini selanjutnya dianonimkan sebagai PT. XYZ, di mana eskalasi infrastruktur *data center* memunculkan rantai hambatan operasional bagi tim SOC meskipun telah mengadopsi Wazuh. Kompleksitas ini memicu empat masalah krusial: (1) fragmentasi operasional akibat arsitektur terdistribusi yang memaksa administrator melakukan navigasi manual berulang ke berbagai *dashboard* untuk memantau ratusan agen; (2) inefisiensi tata kelola kredensial karena ketiadaan *Single Source of Truth* (SSOT) yang menyebabkan data kredensial tidak terpusat dan perubahannya tak terlacak (*untracked*); (3) kesenjangan kontekstual (*contextual gap*) akibat absennya integrasi *Cyber Threat Intelligence* (CTI), sehingga analis harus melakukan *context-switching* untuk verifikasi manual; serta (4) visibilitas pasif tanpa notifikasi *real-time* yang menciptakan *blind spot* saat



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

sistem mengalami *downtime*. Rangkaian kendala ini mengakibatkan alur kerja pertahanan berjalan dalam kondisi suboptimal, di mana sumber daya analis terkuras untuk proses administratif manual alih-alih berfokus pada analisis strategis, yang secara langsung menurunkan kecepatan respon terhadap ancaman.

Menjawab tantangan di PT. XYZ, penelitian ini berfokus pada pembangunan arsitektur dasar bagi ekosistem SOAR (*Security Orchestration, Automation, and Response*) di PT. XYZ. Dengan mengembangkan modul manajemen Wazuh terpusat, sistem notifikasi *real-time*, dan integrasi *Threat Intelligence*, sistem ini berfungsi sebagai "saraf pusat" yang mengintegrasikan langkah-langkah manual ke dalam satu alur kerja terpadu. Infrastruktur ini menjadi elemen fundamental yang melandasi penelitian lanjutan dalam kelompok riset, termasuk pengembangan dasbor *filtering* log terpusat dan modul otomasi respons (*autoblocking*) pada sistem multi-firewall.

Untuk menjamin operasional dengan integritas dan performa tinggi, sistem ini dirancang dengan standar *Online Transaction Processing* (OLTP). Arsitektur ini mengandalkan pola *cache-database* menggunakan PostgreSQL sebagai SSOT untuk menjamin *system of record* yang aman melalui fitur kontrol akses dan enkripsi (Singh, 2015), serta Redis sebagai lapisan *cache in-memory* untuk memastikan responsivitas sistem dalam menangani aliran kueri CTI dari *AlienVault OTX* dan notifikasi Telegram secara instan.

Berdasarkan pendekatan tersebut, penelitian ini mengangkat judul "Optimalisasi Proses Keamanan Proaktif melalui Hub Manajemen Terpusat Multi-Wazuh Terintegrasi *Threat Intelligence* dan *Notifikasi Real-Time*" Implementasi ini diharapkan dapat mentransformasi postur pemantauan keamanan PT. XYZ dari yang bersifat tersebar, pasif, dan manual menjadi terpusat, kontekstual, dan proaktif dengan mengeliminasi langkah-langkah manual yang redundan sehingga lebih optimal.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

1.2 Perumusan Masalah

Berdasarkan latar belakang yang sudah dijabarkan di atas, maka rumusan masalah yang dijadikan fokus pada penelitian ini adalah:

1. Bagaimana merancang bangun Hub Manajemen terpusat sebagai *Single Source of Truth* (SSOT) untuk komponen multi-Wazuh guna mengatasi fragmentasi operasional dan risiko tata kelola kredensial?
2. Bagaimana mengimplementasikan *Threat Intelligence* dan sistem notifikasi *real-time* untuk menutup celah visibilitas (*blind spot*)?
3. Bagaimana mengintegrasikan hub manajemen terpusat dan *threat intelligence* dalam satu platform terpadu guna mengurangi langkah manual dan mengoptimalkan proses pemantauan keamanan proaktif?

1.3 Batasan Masalah

Agar penelitian ini tetap fokus dan terarah pada tujuan yang ingin dicapai, maka penulis menetapkan batasan-batasan masalah sebagai berikut:

1. Pengembangan sistem difokuskan pada manajemen inventaris dan pemantauan status komponen multi-Wazuh (*Manager & Indexer*) secara terpusat beserta informasi Wazuh *Agent*. Penelitian ini tidak mencakup analisis mendalam isi log, *filtering alert*, atau modifikasi komponen.
2. Integrasi komponen dilakukan dengan memanfaatkan *Application Programming Interface* (API) yang disediakan oleh Wazuh, Telegram dan *AlienVault*.
3. Mekanisme notifikasi *real-time* diimplementasikan menggunakan platform Telegram secara satu-arah dan dibatasi pada penyampaian peringatan kondisi operasional sistem.
4. Pengayaan informasi (*enrichment*) menggunakan sumber *open source*, yang secara spesifik dibatasi pada penggunaan *AlienVault* OTX.
5. Evaluasi sistem berfokus pada pengukuran efisiensi operasional (reduksi jumlah langkah kerja dan percepatan waktu akses informasi). Penelitian ini tidak mencakup pengukuran akurasi aturan deteksi (*detection rate*) atau analisis tingkat *false positive*.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

6. Sistem dirancang sebagai aplikasi berbasis web yang berfungsi sebagai antarmuka manajemen administratif (*administrative hub*), bukan sebagai pengganti penuh dasbor visualisasi data historis untuk analisis forensik mendalam.

1.4 Tujuan dan Manfaat

Adapun tujuan yang hendak dicapai melalui penelitian ini meliputi:

1. Merancang bangun sistem manajemen komponen multi-Wazuh terpusat berbasis OLTP dengan arsitektur *cache-database* menggunakan PostgreSQL dan Redis.
2. Mengimplementasikan modul pengayaan informasi melalui integrasi API *Threat Intelligence (AlienVault OTX)* serta notifikasi *real-time* berbasis Telegram sebagai mekanisme peringatan dini guna menutup celah visibilitas (*blind spot*) sistem.
3. Mereduksi jumlah langkah operasional yang dibutuhkan dan mempercepat proses dalam workflow tim keamanan.

Berdasarkan tujuan yang telah dirumuskan, penelitian ini diharapkan dapat memberikan manfaat sebagai berikut:

1. Memangkas waktu yang dibutuhkan tim keamanan dalam mendeteksi dan memvalidasi insiden dengan mengeliminasi langkah-langkah manual (*context-switching*) yang berulang, sehingga membantu organisasi mengejar target waktu respon (*breakout time*) yang optimal.
2. Memberikan referensi akademik mengenai penerapan pola arsitektur *cache-database* (menggunakan PostgreSQL dan Redis) dalam membangun sistem manajemen keamanan siber berkinerja tinggi (*High-Performance Security Management*).
3. Memberikan visibilitas menyeluruh dan konteks ancaman secara instan kepada analis, sehingga meningkatkan kesadaran situasi (*situational awareness*) tanpa perlu melakukan korelasi data manual.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

4. Memitigasi risiko keamanan internal melalui sentralisasi manajemen kredensial (*Single Source of Truth*), memastikan setiap komponen keamanan terdata dan terkontrol dengan baik.
5. Menyediakan infrastruktur dasar yang stabil dan terintegrasi bagi pengembangan fitur keamanan lanjutan, seperti otomasi respons (*autoblocking*) dan analisis log terpusat dalam peta jalan pengembangan SOAR perusahaan.

1.5 Sistematika Penulisan

Berikut adalah sistematika penulisan yang digunakan dalam penyusunan penelitian ini

1. BAB I PENDAHULUAN

Bab ini memaparkan latar belakang yang mendasari penelitian, rumusan masalah, serta batasan masalah agar pembahasan tetap terfokus. Selain itu, dijelaskan pula tujuan dan manfaat yang diharapkan dari penelitian ini.

2. BAB II TINJAUAN PUSTAKA

Bab ini menyajikan landasan teori dan konsep-konsep yang digunakan untuk menganalisis masalah. Di dalamnya juga termuat kajian terhadap penelitian-penelitian terdahulu yang relevan sebagai referensi pendukung.

3. BAB III METODE PENELITIAN

Bab ini merincikan metodologi yang diterapkan, mencakup desain dan tahapan penelitian, objek serta model yang digunakan, teknik pengumpulan dan analisis data, hingga jadwal pelaksanaan penelitian.

4. BAB IV HASIL DAN PEMBAHASAN

Bab ini menguraikan hasil penelitian secara teknis, mulai dari analisis kebutuhan, perancangan dan implementasi sistem, pengujian sistem, hingga analisis terhadap efektivitas sistem yang dibangun.

5. BAB V PENUTUP

Bab terakhir ini berisi kesimpulan yang diperoleh dari seluruh rangkaian penelitian serta saran-saran konstruktif untuk pengembangan penelitian selanjutnya.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

BAB II

TINJAUAN PUSTAKA

2.1 *Security Operations Center (SOC) Proaktif*

Ofoegbu et al. (2024) menegaskan bahwa fokus SOC modern kini beralih dari merespon serangan saat terjadi (reaktif) menuju antisipasi ancaman serangan sebelum terjadi (proaktif). Pergeseran ini menuntut transformasi sistem menjadi dinamis untuk memitigasi ancaman secara *real-time*, tidak lagi sekadar pemantauan pasif (Kumar, 2025). Guna mewujudkannya, integrasi SIEM dan SOAR menjadi strategi proaktif utama (Thakker, 2025). Sebelum adopsi luas teknologi tersebut saat ini, sebelumnya Nagar (2018) telah mengidentifikasi bahwa pendekatan proaktif menuntut inovasi teknologi seperti XDR dan *real-time threat intelligence*. Namun, Grobler et al. (2021) menggarisbawahi bahwa keberhasilan transisi ini tidak hanya bergantung pada penggunaan teknologi, tetapi juga protokol yang berpusat pada pengguna (*user-centric*).

2.2 Efisiensi dan Optimalisasi Alur Kerja SOC

Keberhasilan operasional SOC sangat bergantung pada harmonisasi antara perangkat yang tepat, proses yang baik, serta analis yang bekerja secara efisien dan efektif (Sundaramurthy et al., 2015). Mendukung hal ini, Pulyala, Desetty, dan Jangampet (2019) menyoroti peran SOAR yang mampu mengotomatisasi tugas dan menyederhanakan alur kerja melalui konsep *single pane of glass*. Efisiensi terpusat ini krusial karena memungkinkan analis mengalihkan fokus dari aktivitas repetitif menuju analisis ancaman strategis (Ismail et al., 2025).

Mengacu pada standar ISO 9241-11 (2018), efisiensi didefinisikan sebagai sumber daya yang digunakan sehubungan dengan hasil yang dicapai. Sumber daya utama dalam operasional SOC adalah waktu. Oleh karena itu, peningkatan efisiensi dapat divalidasi secara kuantitatif melalui persentase pengurangan waktu penyelesaian tugas (*Task Completion Time*). Pendekatan ini diperkuat oleh Hyun et al. (2024) yang merumuskan *Deployment Time Reduction Rate (DTRR)* untuk mengukur dampak otomatisasi dengan rumus:

$$DTRR = \frac{\text{Manual deployment time} - \text{Automatic deployment time}}{\text{Manual deployment time}} \times 100\%$$

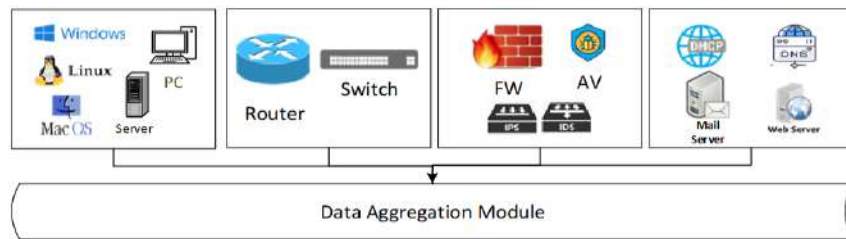


Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

2.3 Security Information and Event Management (SIEM)

Sebagai fondasi pertahanan, Security Information and Event Management (SIEM) bertindak layaknya "sistem saraf pusat" dalam operasi keamanan organisasi (Aare, 2025). Untuk menjalankan fungsi vital ini, Sheeraz et al. (2023) menjelaskan bahwa SIEM bekerja dengan mengagregasi log dari berbagai sumber heterogen seperti *firewall*, *server*, dan *router*, lalu menormalisasi beragam format data tersebut ke dalam satu skema umum. Proses agregasi dan normalisasi ini diilustrasikan secara konseptual pada Gambar 2.2. Unifikasi data ini krusial untuk menyingkap pola serangan yang tidak terlihat jika perangkat hanya dipantau secara terpisah. Dengan demikian, implementasi SIEM memberikan visibilitas jaringan yang komprehensif, memungkinkan deteksi dini terhadap anomali atau aktivitas mencurigakan secara akurat (Sijabat dan Evo, 2023).



Gambar 2.1 Ilustrasi Data Multi-Sumber pada *Data Aggregation Module*

Sumber: Sheeraz et al. (2023)

2.4 Integrasi SOAR, SIEM dan *Firewall*

Security Orchestration, Automation, and Response (SOAR) hadir sebagai solusi transformatif untuk meningkatkan efisiensi operasional keamanan (Pulyala, Desetty, & Jangampet, 2019). Kehadiran teknologi ini menjadi krusial mengingat Kurniawan, Aufar, dan Agustin (2024) menegaskan bahwa pengelolaan sistem secara manual kini menjadi tugas yang tidak efisien dan melelahkan (*laborious*), sebuah tantangan yang dijawab oleh Neumann dan Weber (2021) melalui bukti penurunan MTTD dan MTTR hingga 60% berkat otomatisasi SOAR. Mengintegrasikan kapabilitas ini, Hugo & Proust (2022) mendefinisikan "triad sinergis" terdiri dari *firewall* (penjaga gerbang), SIEM (visibilitas), dan SOAR



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

(orquestrasi) yang membentuk pertahanan holistik untuk deteksi dan respons ancaman proaktif.



Gambar 2.2 Ilustrasi berbagai Kapabilitas SOAR

Sumber: Neumann dan Weber (2021)

2.5 Wazuh

Wazuh dikenal sebagai platform SIEM *open-source* dengan kapabilitas andal dalam deteksi ancaman, pemantauan integritas, dan skalabilitas sistem (Amami, Charfeddine, dan Masmoudi, 2024). Keunggulan ini menunjang efisiensi operasional yang signifikan, di mana Nova, Pratama, dan Prayama (2022) mencatat bahwa dengan Wazuh administrator dapat melakukan pemantauan terpusat secara rutin tanpa terkendala akses fisik ke setiap *server*. Secara teknis, kinerja tersebut didukung oleh arsitektur modular yang memisahkan beban kerja pengumpulan, analisis, dan penyimpanan data.

2.6 Wazuh Agent, Wazuh Manager, dan Wazuh Indexer

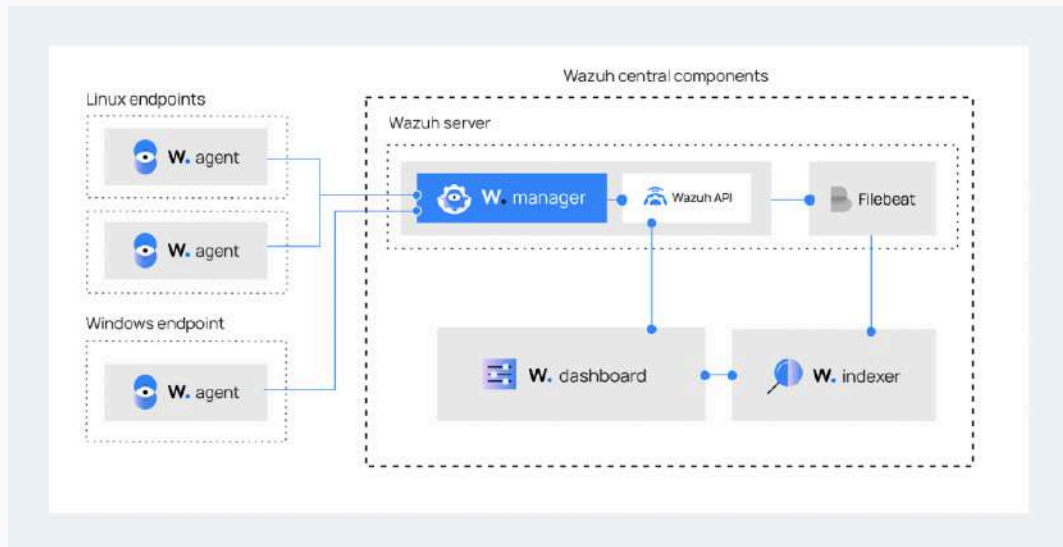
Islam dan Rafique (2024) menguraikan bahwa ekosistem Wazuh terbentuk dari interaksi tiga komponen inti, yaitu Wazuh Agent, Wazuh Manager, dan Wazuh Indexer. Masing-masing komponen memiliki fungsi yang berbeda, tetapi saling terintegrasi dalam proses pengumpulan, pemrosesan, dan penyimpanan data keamanan. Mendalami struktur modular tersebut, Burke (2025) menjelaskan peran serta tanggung jawab dari ketiga komponen utama Wazuh dalam setiap tahapan siklus hidup data keamanan. Hubungan arsitektural serta aliran komunikasi antar



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang menggunakan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

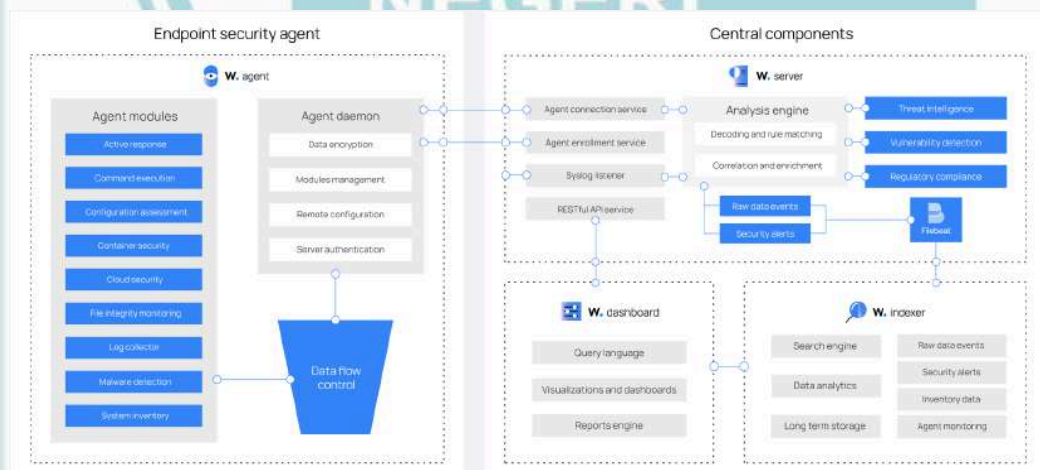
komponen diilustrasikan pada Gambar 2.3 untuk memberikan gambaran yang lebih utuh mengenai cara kerja ekosistem Wazuh.



Gambar 2.3 Topologi Arsitektural Komponen Wazuh

Sumber : (Wazuh, n.d.)

Untuk memberikan pemahaman yang lebih mendalam, Gambar 2.4 menguraikan arsitektur tersebut hingga ke tingkat subkomponen, sehingga interaksi internal, peran setiap layanan pendukung, serta mekanisme komunikasi antarbagian dalam ekosistem Wazuh dapat dipahami dengan lebih jelas.



Gambar 2.4 Fungsi-Fungsi Komponen Wazuh

Sumber : (Wazuh, n.d.)



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritis atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengemukakan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Berdasarkan uraian tersebut, ringkasan karakteristik, fungsi, serta mekanisme akses dari setiap komponen disajikan pada Tabel 2.1.

Tabel 2.1. Perbandingan Karakteristik Komponen Arsitektur Wazuh

Komponen	Fungsi Utama	Port & Protokol Akses	Metode Autentikasi API
Wazuh Agent	Pengumpulan telemetri (<i>Log Collection</i> , FIM) dari <i>endpoint</i> .	1514/TCP (Koneksi ke <i>Manager</i>)	Kunci Enkripsi (<i>Pre-shared Key</i>)
Wazuh Manager	Analisis aturan, korelasi <i>event</i> , dan manajemen agen.	55000/TCP (HTTPS)	Berbasis <i>Token</i> Sesi (JSON Web <i>Token</i>)
Wazuh Indexer	Penyimpanan data jangka panjang dan mesin pencarian log.	9200/TCP (HTTPS)	<i>Basic Authentication</i> (<i>Username & Password</i>)

Sumber: Dirangkum dari Burke (2025) dan Wazuh (n.d.)

2.7 Alienvault OTX sebagai Threat Intelligence

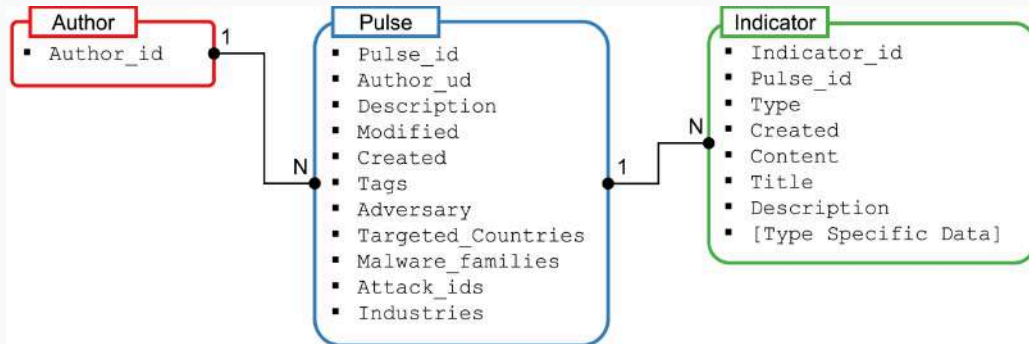
Integrasi *Threat Intelligence* merupakan respons proaktif (Muhammad, Ismail, dan Hassan, 2024) yang menurut Shavaf et al. (2025) mempercepat mitigasi melalui prioritisasi data multi-sumber. Penelitian ini memilih *AlienVault* OTX karena arsitektur Open API dan model data terstrukturnya (Allegretta et al., 2023).

Sebagai platform berbasis komunitas dengan pembaruan kontinu, OTX menyediakan akses via API Key terhadap berbagai *Indicator of Compromise* (IOCs) seperti IP, URL, dan *hash* (Chaitanya et al., 2025). Keunggulan utamanya terletak pada pengayaan konteks; Allegretta et al. (2023) merinci bahwa indikator dilengkapi data geografis, ASN, *WhoIs*, hingga label APT, yang krusial bagi analis untuk memahami ancaman secara instan tanpa penelusuran manual. Gambar 2.5 menunjukkan detail struktur model data *Alienvault* OTX yang mencakup berbagai field untuk Author, *Pulse* dan Indicator.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 2.5 Struktur Model Data *Alienvault OTX*

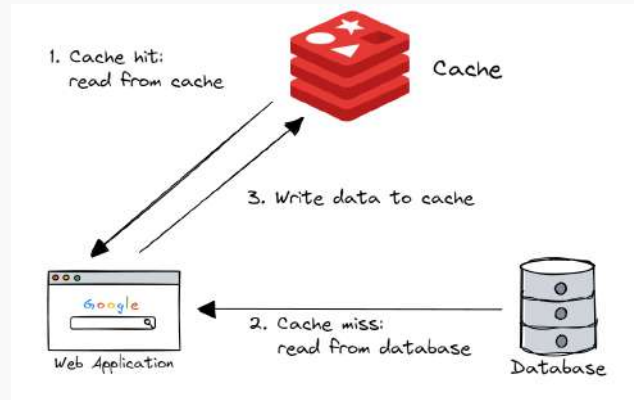
Sumber: Allegretta et al. (2023)

2.8 Arsitektur OLTP *Cache-Database*

Penerapan arsitektur *Online Transaction Processing* (OLTP) pada sistem mengadopsi pola *cache-database* berbasis paradigma *polyglot persistence*. Paradigma ini mengatasi keterbatasan media penyimpanan tunggal dengan menerapkan beberapa teknologi basis data secara bersamaan, di mana masing-masing teknologi dipilih berdasarkan keunggulan spesifiknya terhadap karakteristik data yang dikelola (Barman dan Joshi, 2025).

Pendekatan hibrida tersebut divalidasi oleh Shevchenko (2025), yang menemukan bahwa kolaborasi antara PostgreSQL sebagai lapisan penyimpanan andal dan Redis sebagai cache berkecepatan tinggi mampu mereduksi latensi pengiriman data rata-rata hingga 47% serta menjamin stabilitas operasi selama lonjakan trafik (*traffic spikes*). Kaptsov (2025) menegaskan bahwa pola ini merupakan standar industri yang efektif, di mana aplikasi memprioritaskan pembacaan data dari cache memori untuk memaksimalkan responsivitas antarmuka tanpa mengorbankan integritas data di penyimpanan utama.

Untuk mengimplementasikan strategi ini secara efisien, sistem menggunakan pola *cache-aside* yang merupakan standar umum dalam optimasi akses data (Jovanović, 2024). Alur kerja ini dimulai dengan memeriksa ketersediaan data pada cache Redis; jika terjadi *cache miss*, sistem akan menarik data secara langsung dari sumber utama (*database* persisten), untuk kemudian memperbarui cache tersebut agar permintaan berikutnya dapat dilayani dengan latensi minimal.



Gambar 2.6 Diagram Mekanisme Implementasi *Cache-Database*

Sumber: Jovanović (2024)

2.9 PostgreSQL

Singh (2015) menyoroti bahwa walaupun berbasis *open-source*, PostgreSQL menyediakan fitur keamanan tangguh seperti enkripsi SSL, *Role-Based Access Control*, dan *data masking* yang vital untuk perlindungan data sensitif. Melengkapi aspek keamanan tersebut, PostgreSQL juga menawarkan keunggulan performa signifikan, di mana Salunke dan Ouda (2024) mencatat waktu eksekusinya sekitar 13 kali lebih cepat dibandingkan *MySQL* dengan stabilitas yang terjaga.

2.10 Redis

Berperan sebagai *System of Engagement*, Redis diimplementasikan karena memiliki efisiensi tinggi untuk kebutuhan pemrosesan *real-time* berkat arsitektur *memory-resident* (Andali & Susetyo, 2025). Keunggulan ini didukung oleh temuan Kaptsov (2025), yang menunjukkan mekanisme *caching* Redis mampu memangkas waktu respons rata-rata secara signifikan dari 1.146 ms menjadi 323 ms. Selain fungsi penyimpanan sementara, Redis juga menyediakan mekanisme komunikasi antarkomponen melalui fitur *Publish/Subscribe (Pub/Sub)*. Fitur ini memungkinkan distribusi data secara asinkron melalui *channel* khusus, sehingga data yang dipublikasikan (*publisher*) dapat segera diterima dan diproses oleh komponen yang berlangganan (*subscriber*) tanpa memerlukan mekanisme *polling* secara terus-menerus (Hegde, 2020).

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

2.11 Telegram Bot API

Dharma dan Muslikh (2025) mengidentifikasi bahwa keterlambatan respons sering terjadi karena administrator baru menyadari serangan saat kinerja *server* telah terdegradasi atau mengalami kegagalan total (*down*). Guna memitigasi risiko latensi tersebut, Telegram Bot API diimplementasikan sebagai media penyampaian informasi jarak jauh yang mampu menjamin deteksi aktivitas *server* tersampaikan secara cepat, akurat, dan terdokumentasi (Shafiyyah et al., 2024). Selain kecepatan, pemilihan platform ini didasarkan pada keunggulan teknis berupa tingkat pengiriman (*delivery rate*) yang tinggi serta kapasitas pesan hingga 4.096 karakter. Kapabilitas ini memungkinkan sistem untuk mengirimkan rincian insiden yang mendalam dan komprehensif kepada administrator (Izzatillah, Fania, & Zulfata, 2024).

2.12 Pengujian *Black Box Testing*

Pengujian *Black Box Testing* merupakan metodologi pengujian perangkat lunak yang berfokus pada validasi fungsionalitas sistem tanpa memerlukan pemahaman mendalam terhadap struktur internal atau implementasi kode. Pendekatan ini mengevaluasi apakah setiap fungsi menghasilkan *output* yang sesuai dengan *input* dan kebutuhan yang ditetapkan.

Sebagaimana dijelaskan oleh Pressman dan Maxim (2014), pendekatan ini memvalidasi kesesuaian antara input yang diberikan dengan output atau respons sistem yang dihasilkan untuk memastikan bahwa setiap fitur telah berjalan sesuai dengan persyaratan fungsional yang ditetapkan.

Metode ini dipilih karena efisiensi waktu pengujian yang tinggi, mengingat fokus utama terletak pada perilaku sistem secara eksternal. Untuk menjamin akurasi fungsional, hasil pengujian diukur secara kuantitatif berdasarkan rasio keberhasilan *test case* yang dijalankan terhadap total skenario pengujian yang dirancang. Metrik evaluasi ini ditentukan menggunakan rumus yang dikemukakan oleh Hardika et al. (2024) sebagai berikut:

$$\text{Persentase validasi} = \frac{\text{Jumlah test case valid}}{\text{Total test case}} \times 100\%$$



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

2.13 Transformasi *As-Is* dan *To-Be* untuk Operasional SOC

Analisis proses bisnis *As-Is* dan *To-Be* merupakan instrumen strategis untuk memetakan transformasi operasional di dalam suatu organisasi. Menurut Visual Paradigm (2016), pemodelan *As-Is* memberikan tinjauan menyeluruh terhadap kapabilitas, budaya, dan prosedur yang berjalan saat ini, sementara *To-Be* merumuskan proyeksi ideal dari elemen-elemen tersebut di masa depan. Evaluasi kondisi eksisting (*As-Is*) ini berfokus pada alur kerja yang spesifik di dalam sebuah tim dengan tujuan utama untuk meningkatkan koordinasi dan tingkat responsivitas organisasi (Lucidchart, 2018). Mendefinisikan proses *As-Is* dan *To-Be* juga akan berguna bagi *Security Operations Center* sebagai salah satu bentuk organisasi dengan banyak prosedur alur kerja.

2.14 Standar Parameter Waktu Respon dan Kebutuhan Latensi *Real-Time*

Perancangan infrastruktur keamanan menuntut batasan latensi yang spesifik pada setiap lapisan sistem. Pada lapisan antarmuka (*frontend*), Kelton et al. (2017) menetapkan batas toleransi pengguna (*user's tolerance limit*) maksimal sebesar 5 detik yang menjadi penentu mutlak keberhasilan *User-Perceived Page Load Time* (uPLT). Sementara itu, pada lapisan *backend*, tantangan utamanya bergeser pada efisiensi sumber daya komputasi. Velamoor (2024) memperingatkan bahaya efek *long tail latencies* dalam arsitektur *microservices*. Ketiadaan batas waktu (*timeout*) pada koneksi API yang bersifat asinkron dapat menguras sumber daya *server* secara sia-sia akibat memproses request yang tidak berujung selesai.

Dalam dimensi kecepatan transmisi sangat bergantung pada konteks arsitektural dan kebutuhan sistem, di mana Gault (2025) menegaskan pentingnya pembedaan implementasi antara analitik *true real-time* dan *near-real-time*. Sistem *true real-time* secara kaku memproses data dengan latensi dalam hitungan milidetik, sedangkan sistem *near-real-time* beroperasi dengan *delay* dalam rentang detik hingga menit. Hal ini didukung oleh StreamNative (2025) yang menyatakan bahwa dalam industri data streaming, istilah *real-time* memiliki makna yang bervariasi bergantung pada persyaratan sistem di mana banyak data pipelines yang diklaim sebagai *real-time* di industri secara wajar menoleransi jeda



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

tingkat detik hingga menit; bahkan sistem integrasi sering kali beroperasi optimal dengan latensi 1 hingga 5 menit. Oleh karena itu, parameter *real-time* tidak selalu direpresentasikan oleh kecepatan instan tanpa jeda sama sekali.

Berdasarkan glosarium *real-time data* dari Domo, Inc. (2026), konsep *real-time* didefinisikan sebagai jaminan latensi dari ujung-ke-ujung (*end-to-end latency guarantee*), di mana data dari sumber hingga menjadi sebuah tindakan harus dalam batas *Service-Level Agreement* (SLA) yang telah ditetapkan oleh organisasi. Karakteristik ini sangat berbeda dengan istilah data *live*, yang sekadar mendeskripsikan kondisi antarmuka pengguna (UI) yang melakukan pemutaran ulang (*refresh*) berdasarkan jadwal tertentu dan dapat mengalami penundaan hingga hitungan menit.

Dalam ranah operasional *Security Operations Center* (SOC), SLA *real-time* tersebut memiliki tolak ukur mitigasi yang kritis. Kecepatan pengiriman notifikasi menjadi kunci utama, sebagaimana dibuktikan oleh Siam et al. (2025) yang menetapkan bahwa pengiriman notifikasi kegagalan *firewall* kepada tim SOC harus dieksekusi oleh sistem deteksi *real-time* dalam waktu kurang dari 1 menit. Pada akhirnya, agar seluruh spektrum parameter waktu tersebut terukur secara valid, pengujian harus dilakukan dengan metode yang akurat. Radview (2026) menggarisbawahi pentingnya ketelitian statistik dalam evaluasi kinerja, yang mewajibkan pelaksanaan setidaknya lima iterasi *benchmark* di bawah kondisi operasional yang identik agar simpulan yang ditarik dapat dipertanggungjawabkan keandalannya.

2.15 Diagram Sekuensial dengan Fragmen Kombinasi

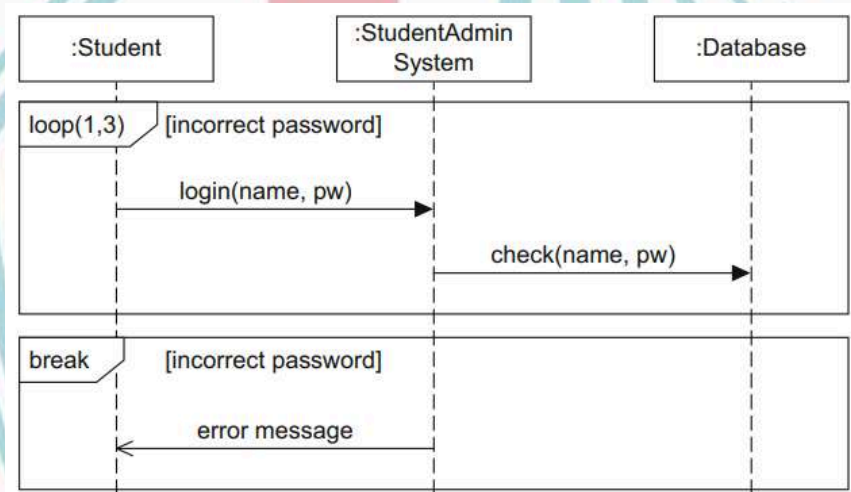
Diagram sekuensial (*sequence diagram*) adalah alat yang sangat berguna bagi staf teknis untuk mendokumentasikan interaksi dalam sistem. Bell (2023) dalam artikelnya pada portal IBM Developer menjelaskan bahwa diagram ini membantu menggambarkan secara jelas bagaimana sistem masa depan seharusnya bekerja. Namun, skenario di dunia nyata sering kali kompleks di mana untuk memodelkan kondisi yang rumit seperti adanya percabangan, pilihan opsional, atau pengulangan diperlukan sebuah mekanisme khusus yang disebut dengan fragmen kombinasi (*combined fragments*).



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Pada dasarnya, fragmen dalam diagram sekuensial UML digunakan untuk menunjukkan pola interaksi yang rumit secara teratur (Seidl et al., 2015). Pola ini mencakup percabangan kondisional, pengulangan, hingga perilaku paralel. Secara visual, fragmen-fragmen tersebut dibungkus dalam sebuah bingkai persegi panjang yang disebut *combined fragment*. Untuk menentukan jenis logika yang berjalan di dalamnya, sebuah operator diletakkan pada sudut kiri atas bingkai tersebut sebagaimana ditunjukkan contoh gabungan alur *loop* dan *break* pada Gambar 2.8 yang membedakan keduanya. Klasifikasi operator ini secara umum dapat dibagi menjadi dua kelompok besar, yakni Branches and loops serta Concurrency and Order.



Gambar 2.8 Contoh Fragment Gabungan *Break* dan *Loop* dalam Satu *Sequential*
Sumber: Seidl et al. (2015)

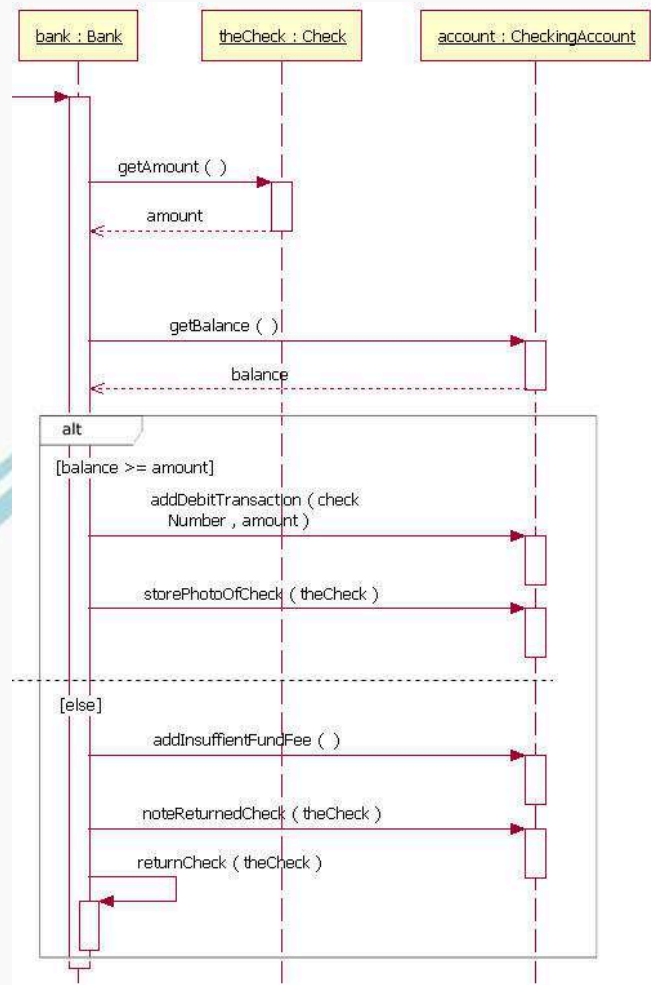
Untuk memodelkan logika program secara akurat pada rancangan sistem usulan, terdapat tiga operator utama yang paling sering digunakan. Merujuk pada dokumentasi Object Management Group (2017), ketiga operator tersebut adalah *'alt'*, *'par'*, dan *'loop'* dengan karakteristik sebagai berikut:

- a. Operator *alt* (*Alternative*): Operator ini mewakili pilihan perilaku kondisional. Di dalam bingkai ini, terdapat kondisi evaluasi (*guard condition*) di mana paling banyak hanya satu pilihan jalur (*operand*) yang akan dieksekusi oleh sistem. Jika salah satu kondisi terpenuhi, pilihan jalur yang lain akan diabaikan.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 2.9 Contoh Diagram Sekuensial dengan Alternatives

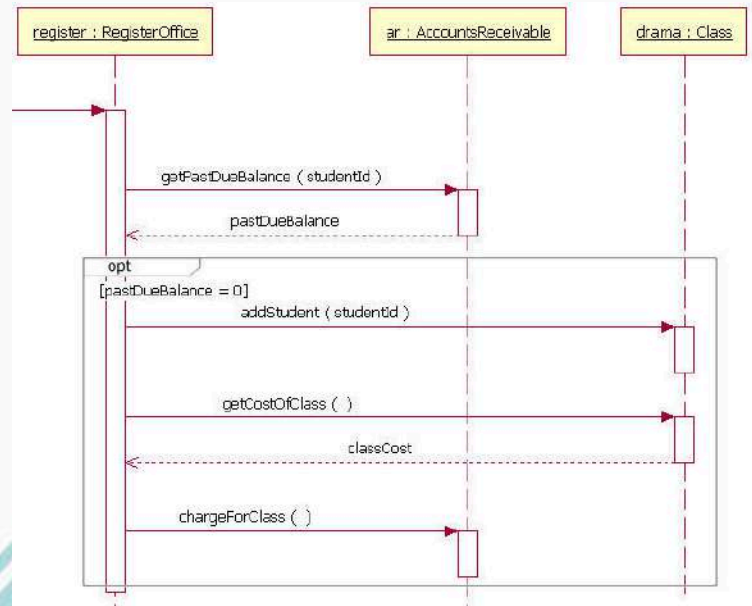
Sumber: Bell (2023)

- b. Operator *par* (Parallel): Operator ini menunjukkan bahwa beberapa interaksi dapat berlangsung secara paralel dalam waktu yang bersamaan. Setiap jalur eksekusi berjalan secara independen sehingga tidak bergantung pada penyelesaian jalur lainnya untuk dapat dimulai. Dalam kondisi ini, urutan kejadian antar jalur dapat saling tumpang tindih (*interleaved*), selama tidak melanggar ketergantungan yang telah ditentukan. Dengan demikian, operator *par* digunakan untuk merepresentasikan aktivitas yang dapat dieksekusi secara bersamaan, sehingga mencerminkan perilaku sistem yang lebih realistis ketika beberapa proses berlangsung secara simultan.



Hak Cipta :

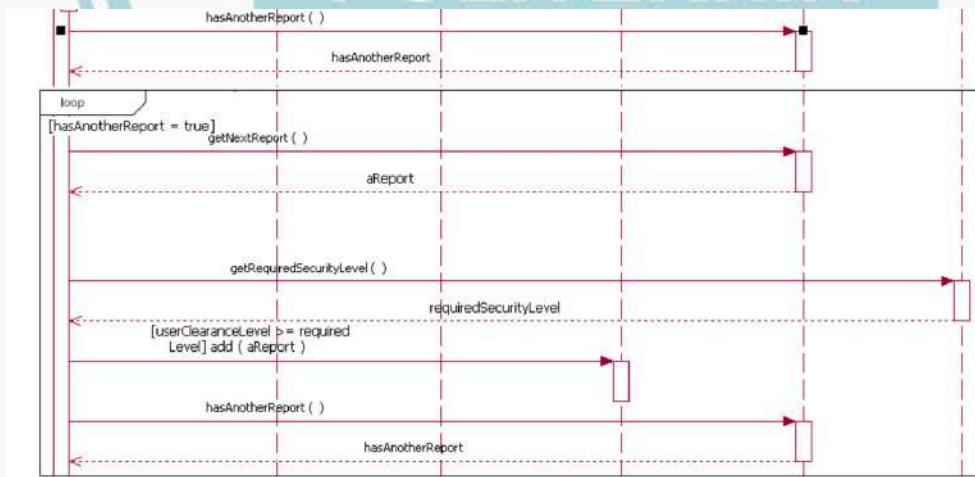
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 2.10 Contoh Diagram Sekuensial dengan Operator Parallel

Sumber: Bell (2023)

- c. Operator *loop* (*looping*): Operator ini merepresentasikan proses pengulangan. Seluruh instruksi atau pesan di dalam bingkai ini akan dieksekusi secara berulang kali sesuai dengan batas minimum dan maksimum yang sudah ditentukan oleh sistem.



Gambar 2.11 Contoh Diagram Sekuensial dengan Operator *Loop*

Sumber: Bell (2023)



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Meskipun struktur alur logika pada diagram sekuensial ini dikendalikan oleh standar *Combined Fragments* UML (seperti *alt*, *par*, dan *loop*), aspek visualisasi pada diagram turut disesuaikan untuk mengoptimalkan pemahaman pembaca. Berdasarkan spesifikasi Object Management Group (2017), UML menyediakan elemen *Comment* serta kebebasan opsi presentasi, di mana penggunaan warna dan anotasi tekstual tambahan diizinkan selama tidak mengubah semantik logika inti.

2.16 Penelitian Terkait

Sebagai landasan pijak penelitian, Tabel 2.2 menyajikan pemetaan penelitian-penelitian terdahulu (*State of the Art*) yang relevan dengan topik yang dibahas. Pemetaan tersebut tidak hanya memberikan gambaran mengenai perkembangan penelitian yang telah dilakukan, tetapi juga menyoroti perbedaan signifikan pada fitur, metode, serta pendekatan yang ditawarkan dalam penelitian ini dibandingkan dengan penelitian sebelumnya.

Tabel 2.2. Perbandingan Penelitian Terkait

No	Judul dan Penulis	Kata Kunci dan Hasil	Perbedaan
1	<i>Hybrid Multilayer Architecture Integrating Suricata, Wazuh, and Cyber Threat Intelligence for Drive-by-Download Malvertising Detection</i> (Adrian et al., 2026)	○ Kata Kunci: Wazuh, Suricata, IDS, VirusTotal, DDoS. ○ Hasil: Mengintegrasikan Suricata, Wazuh, dan VirusTotal untuk deteksi dan respon otomatis terhadap serangan <i>drive-by-download</i>. Sistem berhasil memitigasi ancaman dalam 5-7 detik dengan akurasi tinggi melalui korelasi multi-layer antara SIEM, IDS dan CTI.	○ Penelitian ini berfokus murni pada logika deteksi dan aliran data antar-komponen dalam lingkungan isolasi (VM), namun tidak membangun Managerial Hub terpusat untuk tata kelola kredensial (<i>credential governance</i>) dan manajemen aset yang krusial bagi skalabilitas operasional. ○ Penelitian penulis mengisi celah tersebut dengan membangun sistem manajemen berbasis arsitektur <i>containerized</i>, menjawab tantangan <i>future work</i> dari penelitian ini terkait evaluasi pada lingkungan <i>real-world deployment</i>.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengemukakan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

No	Judul dan Penulis	Kata Kunci dan Hasil	Perbedaan
2	<i>Architectural Framework for Threat Intelligence Integration with SIEM and SOAR in Hybrid Cloud Security Environments</i> (Kummarapurugu, 2024)	○ Kata Kunci: <i>Threat Intelligence</i>, SIEM, SOAR, <i>Automated Response</i>. ○ Hasil: Merumuskan kerangka kerja arsitektural (<i>framework</i>) yang menggabungkan aliran data dari SIEM, pengayaan <i>Threat Intelligence</i>, dan otomasi SOAR. Penelitian ini memvalidasi secara teoritis bahwa integrasi ketiga komponen ini secara signifikan meningkatkan visibilitas ancaman dan mempercepat waktu respons insiden di lingkungan <i>hybrid</i>.	○ Penelitian ini berfokus pada pengembangan kerangka kerja konseptual (<i>architectural framework</i>) dan metodologi integrasi secara umum. ○ Penelitian penulis menerapkan konsep tersebut menjadi implementasi teknis nyata berupa Hub Manajemen Terpusat yang berfungsi sebagai <i>Single Source of Truth</i> (SSOT) untuk pengelolaan komponen Multi-Wazuh dan integrasi <i>Threat Intelligence</i> otomatis, guna menyediakan infrastruktur yang mendukung proses kerja SOAR.
3	Sistem Monitoring Keamanan Pada <i>Data Center</i> Berbasis SIEM Dengan Wazuh dan IDS (Matin et al., 2025)	○ Kata Kunci: Wazuh, Suricata, IDS, Port Scanning, <i>DDoS</i>. ○ Hasil: Menggabungkan Wazuh dan Suricata dalam satu <i>server</i> untuk mendeteksi serangan jaringan seperti port scanning dan <i>DoS</i>. Sistem berhasil menampilkan log serangan jaringan pada <i>dashboard</i> bawaan Wazuh.	○ Penelitian sebelumnya berfokus pada pemantauan dalam lingkup satu <i>server</i> (<i>single-node</i>) yang digabungkan dengan IDS. ○ Penelitian penulis mengembangkan sistem manajemen terpusat yang mampu mengelola banyak <i>server</i> Wazuh (Multi-Wazuh) sekaligus dalam satu platform untuk mengatasi masalah pemantauan yang terpecah-pecah.
4	Experimental Evaluation of Wazuh-Grafana Integration for <i>Real-Time Cyber Threat Detection in Resource-Constrained Environments</i> (Sutanto & Rakhman, 2025)	○ Kata Kunci: Wazuh, Grafana, <i>Real-time</i> SIEM, <i>Resource-Constrained</i>. ○ Hasil: Mengintegrasikan Wazuh dengan Grafana untuk meningkatkan visualisasi deteksi pada lingkungan dengan sumber daya terbatas (VPS). Hasilnya adalah akurasi deteksi 100% dan visualisasi <i>real-time</i>	○ Penelitian sebelumnya hanya memecahkan masalah visualisasi data pada satu <i>instance server</i>. Tidak ada mekanisme kontrol manajemen (<i>control plane</i>) untuk mengelola inventaris jika terdapat banyak <i>server</i> Wazuh. ○ Penelitian penulis membangun <i>Administrative Hub</i> yang berfungsi sebagai <i>Single Source of Truth</i> (SSOT) untuk mengontrol dan memantau



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

No	Judul dan Penulis	Kata Kunci dan Hasil	Perbedaan
		dengan penggunaan sumber daya rendah.	status kesehatan (<i>health monitoring</i>) komponen multi-Wazuh secara terpusat.
5	Implementasi Wazuh Menggunakan Metode PPDIOO di Sistem Keamanan Jaringan PSDKU Universitas Lampung Waykanan Sebagai Deteksi dan Respon Serangan Siber (Shafiyyah et al., 2024)	○ Kata Kunci: Wazuh, PPDIOO, Fail2ban, Telegram, Brute Force. ○ Hasil: Mengimplementasikan Wazuh untuk mendeteksi serangan Brute Force dan DoS. Mekanisme respons menggunakan Fail2ban untuk memblokir IP penyerang di level <i>host</i> dan mengirimkan notifikasi via Telegram.	○ Penelitian terdahulu menerapkan pemantauan pada satu <i>server</i> Wazuh tunggal. Notifikasi yang dikirimkan berupa <i>raw alert</i> namun bukan sebagai notifikasi terkait kondisi sistem Wazuh. ○ Berbeda dengan penelitian penulis yang membangun sistem terpusat untuk memonitor kesehatan banyak komponen Wazuh secara bersamaan, serta opsi pengayaan dengan <i>Threat Intelligence (AlienVault)</i> agar analisis bisa mendapatkan konteks ancaman.

POLITEKNIK
NEGERI
JAKARTA

BAB III

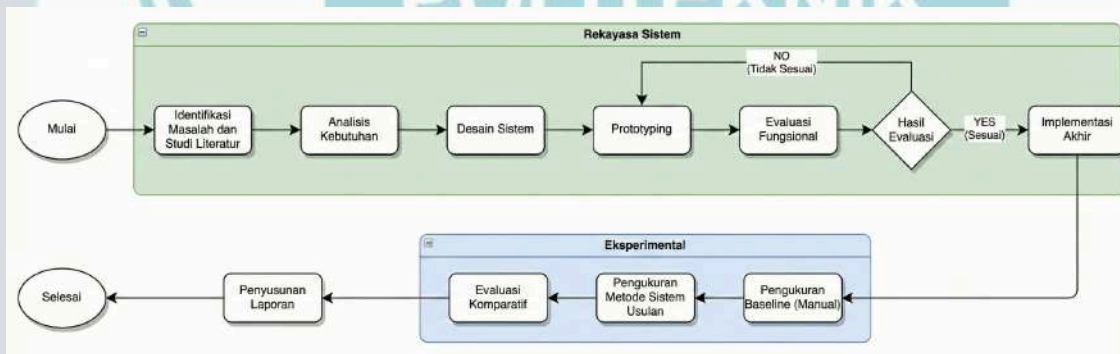
METODE PENELITIAN

3.1 Rancangan Penelitian

Penelitian ini menerapkan metode Rekayasa Sistem (*System Engineering*) dengan pendekatan Eksperimental. Pemilihan metode rekayasa sistem didasari oleh karakteristik masalah yang dihadapi, yakni kebutuhan untuk membangun Hub Manajemen Terpusat yang mengintegrasikan berbagai API (*Application Programming Interface*) dari sistem yang berbeda (Wazuh, Telegram dan *Threat Intelligence*). Dalam metode rekayasa sistem akan digunakan model *prototyping* memungkinkan validasi fungsi integrasi secara bertahap dan perbaikan iteratif sebelum sistem diuji secara menyeluruh.

Sementara itu, pendekatan Eksperimental digunakan untuk menjawab rumusan masalah terkait optimalisasi. Pendekatan ini dilakukan dengan membandingkan kinerja operasional (variabel waktu dan jumlah langkah) antara metode manual (*existing*) dengan metode menggunakan sistem yang diusulkan.

Alur penelitian dirancang secara sistematis untuk memastikan setiap tahapan berkontribusi langsung pada pencapaian tujuan optimalisasi proses keamanan. Visualisasi alur penelitian dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram Tahapan Penelitian

3.2 Tahapan Penelitian

Berdasarkan alur rancangan di gambar 3.1, tahapan penelitian dilaksanakan dengan rincian sebagai berikut dari tahapan awal identifikasi masalah dan studi literatur, desain sistem berupa *prototyping* hingga implementasi akhir dan penyusunan laporan:



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

1. Identifikasi Masalah dan Studi Literatur:

Tahap awal untuk memetakan kendala operasional seperti fragmentasi data pada multi-Wazuh dan inefisiensi validasi ancaman manual. Bersamaan dengan itu, dilakukan studi terhadap dokumentasi teknis API Wazuh, *AlienVault OTX*, serta arsitektur *microservices* sebagai landasan solusi.

2. Analisis Kebutuhan Sistem:

Menentukan spesifikasi fungsional sistem, yaitu kemampuan sistem untuk bertindak sebagai *Single Source of Truth* (SSOT) kredensial, melakukan penarikan data status agen secara *real-time*, dan menyediakan mekanisme *lookup* data intelijen secara terpusat.

3. Desain Sistem:

Merancang arsitektur teknis sebelum implementasi, meliputi desain skema basis data PostgreSQL untuk konfigurasi, mekanisme *caching* Redis untuk performa, serta rancangan antarmuka *dashboard* tunggal.

4. Pembangunan *Prototype* (*Prototyping*):

Tahap implementasi kode (*coding*) untuk mewujudkan rancangan menjadi perangkat lunak yang dapat dijalankan. Pada tahap ini, fokus utama adalah membangun jalur komunikasi data (*data pipeline*), modul konektor API (*Ingestion Layer*) serta web *interface* untuk hub Wazuh, *threat intelligence* dan notifikasi.

5. Evaluasi *Prototype*:

Dilakukan pengujian fungsional (*Black Box Testing*) terhadap *prototype* yang terbangun. Berlanjutnya penelitian dari fase ini ke fase lainnya ditentukan oleh:

- Jika Tidak Sesuai (terdapat *bug* atau kegagalan integrasi API), proses kembali ke tahap *Prototyping* untuk perbaikan kode (*refinement*).
- Jika Sesuai (fungsi berjalan stabil), proses berlanjut ke tahap berikutnya.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

6. Implementasi Akhir:

Penerapan versi stabil dari Hub Manajemen ke dalam lingkungan pengujian (*staging*) yang telah dikonfigurasi dengan *server* Wazuh aktif, siap untuk pengambilan data.

7. Pengukuran *Baseline* (Manual):

Melakukan pengukuran kinerja operasional menggunakan metode eksisting (manual). Penulis mencatat waktu tempuh (*time duration*) dan jumlah langkah (*step count*) yang dibutuhkan analis untuk memonitor dan memvalidasi insiden menggunakan *dashboard* bawaan Wazuh yang terpisah-pisah, dan untuk *lookup threat intelligence*.

8. Pengukuran Metode Sistem Usulan:

Melakukan pengukuran ulang dengan skenario tugas yang sama, namun menggunakan Hub Manajemen Terpusat. Data waktu dan langkah dicatat untuk dibandingkan.

9. Evaluasi Komparatif:

Menganalisis selisih data antara pengukuran *baseline* dan sistem usulan. Tahap ini bertujuan menghitung persentase efisiensi (reduksi waktu dan langkah) yang dihasilkan oleh sistem.

10. Penyusunan Laporan:

Mendokumentasikan seluruh proses perancangan, hasil pengujian, dan analisis data ke dalam laporan akhir skripsi.

3.3 Objek Penelitian

Objek penelitian ini adalah Hub Manajemen Keamanan Terpusat yang berfungsi sebagai modul orkestrasi utama dalam sistem Security Orchestration, Automation, and Response (SOAR).

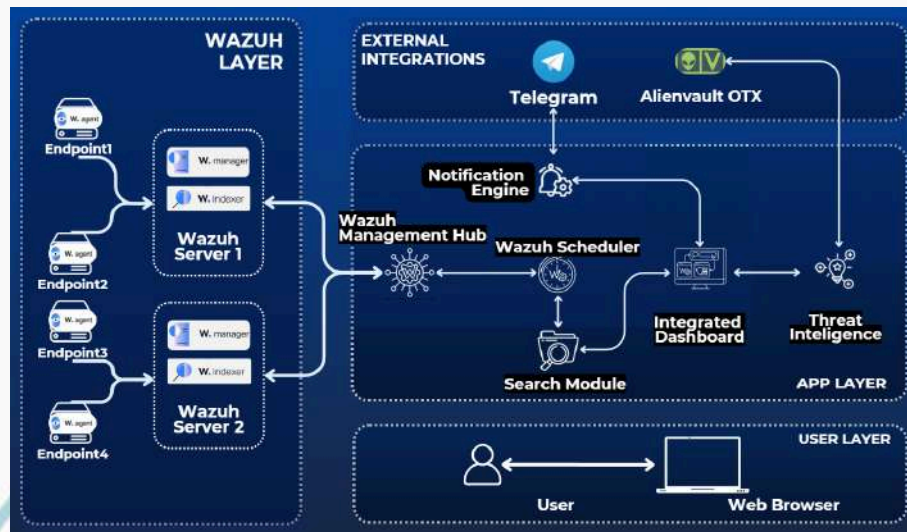
Ruang lingkup sistem dalam penelitian ini dirancang pada lapisan aplikasi (*Proposed App Layer*) untuk mengintegrasikan lapisan deteksi (*Wazuh Layer*) dengan layanan informasi eksternal (*External Integrations*). Secara spesifik, komponen yang dikaji meliputi modul Wazuh Management Hub (untuk sentralisasi status dan inventaris), integrasi *Threat Intelligence*, dan *Notification Engine*. Gambaran umum arsitektur sistem disajikan pada Gambar 3.2, di mana



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

bagian yang ditandai merepresentasikan komponen spesifik yang dikembangkan dalam penelitian ini.



Gambar 3.2 Diagram Arsitektur Objek Penelitian

Spesifikasi deskriptif mengenai objek penelitian adalah sebagai berikut:

1. Fungsi Utama: Menyediakan Hub Manajemen Terpusat untuk memantau status konektivitas dan inventaris agen dari berbagai *instance* Wazuh secara *real-time*, memfasilitasi validasi ancaman (*Threat Intelligence*) secara *on-demand*, serta mengirimkan notifikasi status operasional sistem.
2. Pengguna Sistem: *SOC Analyst Executor* (untuk pemantauan status dan validasi insiden) dan *System Administrator* (untuk manajemen koneksi API dan integrasi).
3. Input Data: Data status operasional (*health check*) dan daftar agen yang ditarik melalui API Wazuh, data reputasi ancaman dari AlienVault OTX, serta parameter konfigurasi koneksi sistem.
4. Teknologi Inti: Menggunakan PostgreSQL (*OLTP Database*), Redis (*In-memory store*) dan Python (Flask) sebagai kerangka kerja *backend*, dan HTML, CSS serta Javascript sebagai kerangka kerja *frontend*.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

BAB IV

HASIL DAN PEMBAHASAN

4.1 Analisis Kebutuhan

Tahap analisis kebutuhan merupakan landasan untuk menentukan spesifikasi teknis dari sistem yang akan dibangun. Berdasarkan observasi pada lingkungan operasional PT. XYZ, tim *Security Operations Center* (SOC) saat ini harus mengelola infrastruktur terfragmentasi skala besar yang terdiri dari 4 *node Wazuh Indexer*, 4 *node Wazuh Manager* yang memiliki sejumlah *Wazuh Agent* masing-masing. Guna menopang beban operasional pada skala tersebut, analisis kebutuhan sistem dalam penelitian ini dirumuskan secara komprehensif ke dalam empat aspek utama, yaitu: kebutuhan fungsional, kebutuhan non-fungsional, kebutuhan perangkat keras (*hardware*), dan kebutuhan perangkat lunak (*software*).

4.1.1 Kebutuhan Fungsional (Functional Requirements)

Kebutuhan fungsional mendefinisikan apa yang harus dapat dilakukan oleh sistem. Berdasarkan ruang lingkup penelitian, kebutuhan fungsional dibagi ke dalam tiga modul utama:

A. Modul Hub Manajemen Terpusat Multi-Wazuh (*Single Source of Truth*)

Modul ini berfungsi sebagai pusat kendali untuk mengelola dan memantau seluruh komponen Wazuh yang terdistribusi. Kebutuhan fungsionalnya meliputi:

1. Sistem mampu mengonfigurasi koneksi berbagai *Wazuh Manager* dan *Wazuh Indexer*, yang mencakup kemampuan bagi Administrator untuk menambahkan, melihat, mengubah, dan menghapus data.
2. Sistem mampu menampilkan daftar inventaris agen (*Agent List*) dari seluruh *Wazuh Manager* yang terdaftar secara terpadu di dalam satu dasbor utama.
3. Sistem mampu menampilkan informasi status operasional terkini (*Online/Offline*) dari setiap komponen Wazuh secara visual kepada pengguna.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

4. Sistem mampu memberikan informasi validasi (umpan balik keberhasilan atau kegagalan) terkait kebenaran kredensial komponen Wazuh yang sedang didaftarkan oleh pengguna.

B. Modul Pengayaan *Threat Intelligence*

Modul ini bertugas menyediakan pengayaan data (*data enrichment*) untuk memvalidasi indikator kompromi (IoC) tanpa harus berpindah aplikasi. Kebutuhan fungsionalnya meliputi:

1. Sistem mampu mengelola kunci autentikasi (*API Key*) layanan *Threat Intelligence* (AlienVault OTX) yang dikonfigurasi oleh Administrator.
2. Sistem mampu mengakomodasi pencarian dan validasi Indikator Kompromi (IoC) spesifik, seperti Alamat IP (IPv4) atau *File Hash*, yang dimasukkan oleh Analis.
3. Sistem mampu memproses dan menarik data pengayaan ancaman (*threat enrichment*) dari penyedia layanan eksternal secara langsung saat analisis dibutuhkan.
4. Sistem mampu menyajikan hasil analisis meliputi status reputasi, geolokasi, dan riwayat ancaman terkait (*threat pulses*) dalam format visual yang terstruktur dan mudah dipahami oleh pengguna.

C. Modul Notifikasi *Real-Time* (Telegram)

Modul ini dirancang sebagai mekanisme peringatan dini (*early warning system*) jika terjadi anomali pada ketersediaan infrastruktur, dengan kebutuhan fungsional sebagai berikut:

1. Sistem mampu mengelola data konfigurasi distribusi notifikasi, yang mencakup penyimpanan Bot *Token* dan daftar identitas tujuan pesan (*Chat ID*) Telegram.
2. Sistem mampu memantau status kesehatan (*health status*) dan ketersediaan dari infrastruktur Hub maupun *node* Wazuh secara otomatis dan berkelanjutan.
3. Sistem mampu melakukan pengiriman notifikasi testing.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

4.1.2 Analisis Kebutuhan Non-Fungsional

Kebutuhan non-fungsional mendefinisikan kriteria kualitas teknis dan batasan operasional sistem. Mengingat arsitektur sistem ini mengintegrasikan tiga layanan dengan karakteristik protokol dan letak jaringan yang berbeda, kebutuhan non-fungsional diuraikan secara spesifik untuk masing-masing fungsi inti:

A. Kebutuhan Kinerja dan Waktu Respons (Performance & Latency)

Tingkat toleransi waktu respons (*response time*) disesuaikan dengan sumber data masing-masing modul:

1. Kinerja Hub Multi-Wazuh: Operasi *Read* untuk memuat halaman *Dashboard* dan agregasi status multi-*node* Wazuh harus diselesaikan dalam waktu di bawah 5 detik. Target ini dicapai dengan memanfaatkan mekanisme *caching in-memory* (Redis) agar sistem tidak perlu melakukan *query* langsung ke setiap *server* Wazuh setiap kali page di-*refresh*.
2. Kinerja *Threat Intelligence*: Mengingat pencarian data bergantung pada latensi jaringan internet publik ke *server* AlienVault OTX, sistem wajib menerapkan proses asynchronous (misalnya AJAX). UI tidak boleh *freeze*, melainkan menampilkan indikator loading, dengan batas waktu tunggu maksimal (*timeout request*) diset pada 15 detik sebelum sistem mengembalikan pesan gagal ke pengguna.
3. Kinerja Notifikasi Telegram: Proses deteksi perubahan status hingga trigger pengiriman pesan via Telegram harus dapat dieksekusi oleh background *scheduler* dengan latensi maksimal 1 menit (tergantung siklus/interval pengecekan *health check* latar belakang).

B. Kebutuhan Keamanan Data Sensitif (Security & Encryption)

Sebagai *Single Source of Truth* (SSOT), sistem menyimpan berbagai tipe kredensial (*secrets*) yang memiliki tingkat sensitivitas dan metode autentikasi yang berbeda. Semua *secrets* wajib dienkripsi secara simetris (*Encryption at Rest*) di dalam *database* dengan spesifikasi peruntukan:



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

1. Keamanan Modul Wazuh: Sistem harus mengenkripsi kombinasi *Username* dan *Password* (atau *JWT Credentials*) dari setiap *instance* Wazuh yang didaftarkan.
2. Keamanan *Threat Intelligence*: Sistem harus mengenkripsi *API Key* tunggal milik penyedia layanan AlienVault OTX yang dikonfigurasi oleh Administrator.
3. Keamanan Modul Telegram: Sistem harus mengenkripsi *Bot Token* Telegram (yang berfungsi sebagai kunci akses bot). Sementara itu, *Chat ID* (tujuan pengiriman) dapat disimpan dalam format *plaintext* biasa karena tidak bersifat *executable*.
4. Keamanan Akses Sistem (*Global*): *Password* pengguna Hub harus di-hash dan manajemen sesi (*Session Management*) wajib menggunakan atribut *Secure* dan *HttpOnly cookie* untuk mencegah peretasan sesi.

C. Kebutuhan Keandalan dan Toleransi Kesalahan (*Reliability & Fault Tolerance*)

Kebutuhan ini mendefinisikan kemampuan sistem untuk mempertahankan stabilitas sistem saat menghadapi anomali jaringan atau gangguan layanan eksternal. Sistem harus mampu mengisolasi gangguan tersebut dan tetap menjalankan fungsi utama yang tersisa secara optimal dengan kriteria sebagai berikut:

1. Toleransi *Node* Wazuh: Jika satu *server* Wazuh mengalami gangguan, *dashboard* Hub harus tetap berfungsi *normal* dan tetap menampilkan data dari *server* Wazuh lain yang masih hidup, dengan memberikan label visual "*Offline*" pada *node* yang bermasalah.
2. Penanganan Kesalahan Integrasi Eksternal (*Graceful Error Handling*): Jika sistem gagal terhubung ke API AlienVault OTX (misalnya akibat *API Key* tidak valid atau layanan terputus), antarmuka pengguna tidak boleh *freeze* atau mengekspos *error log backend*. Sistem harus mengembalikan pesan kesalahan yang informatif kepada analis.
3. Ketahanan Modul Notifikasi terhadap Kesalahan Kredensial: Apabila terjadi kesalahan autentikasi pada layanan Telegram (misalnya *Bot Token*



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

tidak valid atau *Chat ID* salah), sistem harus mampu mengisolasi kegagalan tersebut. Proses gagalnya pengiriman pesan tidak boleh menyebabkan putusnya pemantauan latar belakang, melainkan hanya dicatat sebagai eksepsi pada *error log* sistem.

4.1.3 Analisis Kebutuhan Perangkat Keras (*Hardware*)

Infrastruktur *backend* (Aplikasi Hub, PostgreSQL dan Redis) di-*deploy* pada sebuah *Virtual Private Server* (VPS) berbasis XCP-ng yang difasilitasi oleh mitra industri, yaitu PT. XYZ. Berikut adalah rincian spesifikasi perangkat keras yang dialokasikan:

Tabel 4.1. Daftar Kebutuhan Perangkat Keras (*Hardware*)

No	<i>Hardware</i>	Deskripsi	Spesifikasi
1	Prosesor (CPU)	Mendukung komputasi paralel dan OLTP	16 vCPU Intel® Xeon® CPU E5-2630 v4 @ 2.20GHz (Arsitektur 64-bit)
2	Memori (RAM)	Mengakomodasi kebutuhan memori utama untuk <i>caching</i> serta operasional <i>in-memory data store</i> Redis tanpa memicu terjadinya <i>memory swapping</i> .	16 GB RAM
3	Penyimpanan (Storage)	Menjamin nilai IOPS (<i>Input/Output Operations Per Second</i>) yang tinggi pada saat proses <i>ingestion</i> data secara <i>real-time</i> dan pencarian log pada basis data.	<i>Solid State Drive</i> (SSD) / NVMe



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

4	Jaringan (Network)	Menyediakan konektivitas pita lebar dengan latensi rendah ke <i>node</i> Wazuh dan internet publik untuk memastikan penarikan status dan pengiriman notifikasi berjalan presisi.	<i>Virtualized Ethernet Interface</i> (vif)
---	-------------------------------	--	---

4.1.4 Analisis Kebutuhan Perangkat Lunak (*Software*)

Sistem Hub Manajemen Terpusat dirancang menggunakan pendekatan arsitektur *microservices* yang modern. Kebutuhan perangkat lunak pendukung diuraikan pada tabel berikut:

Tabel 4.2. Daftar Kebutuhan Perangkat Lunak (*Software*)

No	<i>Software</i>	Deskripsi	Versi
1	Ubuntu Server	Sistem operasi lingkungan <i>server</i> berbasis Linux (<i>headless</i>) untuk stabilitas pengelolaan arsitektur jaringan dan efisiensi <i>resource</i> .	Ubuntu 22.04.5 LTS (Kernel 5.15.0-151) 64-bit
2	Wazuh (Manager, Dashboard ,Indexer)	Bertindak sebagai mesin deteksi utama yang memonitor dan mengumpulkan peringatan keamanan (security alerts).	Wazuh v4.14.3
3	Docker & Docker Compose	Mesin kontainerisasi untuk membungkus aplikasi dan <i>database</i> agar terisolasi, mencegah konflik dependensi, dan memudahkan <i>deployment</i> .	Docker <i>Engine</i> v28.2.2 & Docker Compose v2.36.2



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

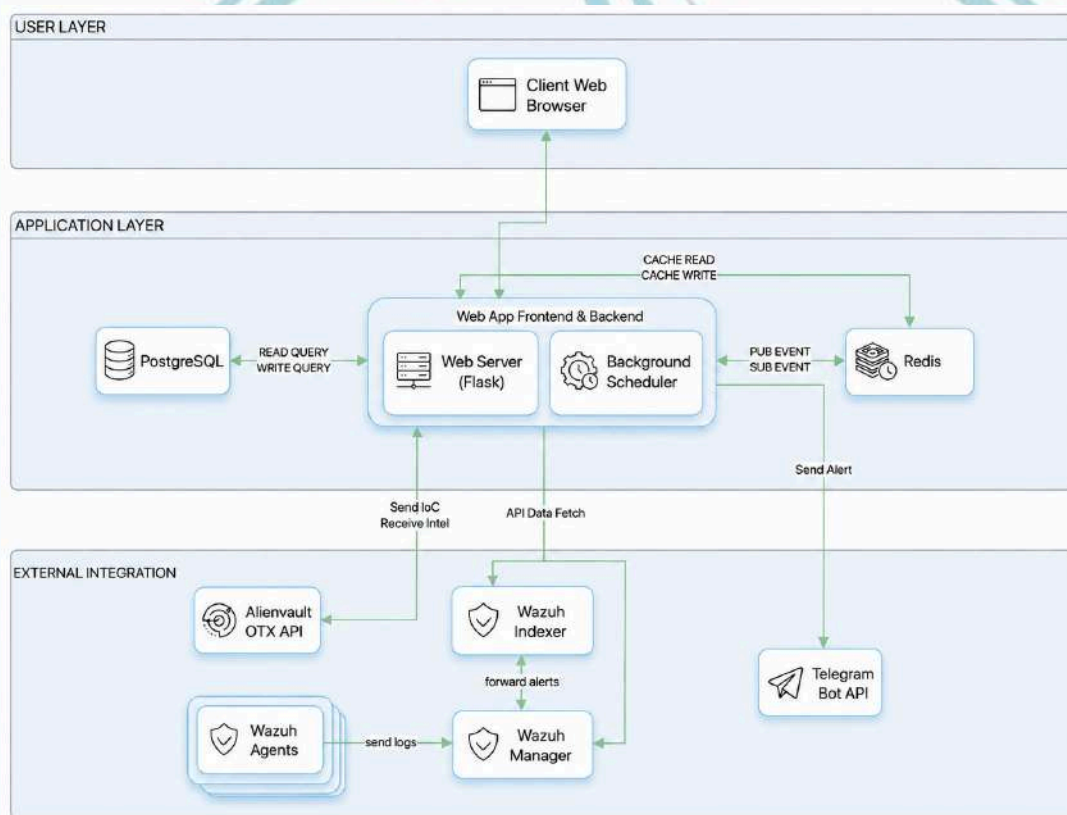
4	Python & Flask	Bahasa pemrograman dan kerangka kerja <i>backend (micro-framework)</i> untuk membangun RESTful API <i>Gateway</i> dan skrip <i>scheduler</i> latar belakang.	Python 3.11.14, Flask v3.1.2
5	PostgreSQL	<i>Relational Database Management System</i> (RDBMS) untuk menyimpan data konfigurasi kredensial (SSOT) pengguna secara terstruktur dan terenkripsi.	PostgreSQL v15.13
6	Redis	<i>In-memory data store</i> yang berfungsi sebagai <i>cache</i> performa tinggi untuk antarmuka pengguna dan <i>Message Broker</i> (Pub/Sub) untuk antrian notifikasi.	Redis Server v7.4.5
7	Postman	Perangkat lunak uji coba untuk memvalidasi <i>request</i> dan <i>response payload</i> JSON dari API Wazuh maupun API <i>AlienVault OTX</i> .	<i>Latest Build</i>
8	Database Client JDBC	Alat manajemen <i>database</i> GUI untuk memverifikasi rancangan skema dan keberhasilan fungsi enkripsi (<i>encryption at rest</i>) pada tabel.	v1.4.6
9	Web Browser	Antarmuka interaktif bagi pengguna (<i>Analisis/Admin</i>) untuk mengakses <i>Unified Dashboard</i> tanpa memerlukan instalasi perangkat lunak tambahan (<i>clientless</i>).	Chrome / Firefox (Mendukung HTML5, CSS3, ES6+)



4.2 Perancangan Sistem

4.2.1 Arsitektur Perangkat Lunak dan Aliran Data

Perancangan arsitektur untuk Hub Manajemen Keamanan Terpusat ini dibangun menggunakan pendekatan *microservices* yang dipadukan dengan pola *cache-database* (OLTP). Arsitektur ini dirancang untuk memisahkan beban kerja komputasi antara interaksi pengguna (*synchronous*) dan proses pemantauan otomatis (*asynchronous*). Setiap komponen dan lapisan dalam arsitektur hub ini digambarkan seperti diagram Gambar 4.1.



Gambar 4.1 Diagram Arsitektur Hub Manajemen Keamanan Terpusat

Secara spesifik, arsitektur sistem dirancang dan dibagi ke dalam empat lapisan utama yang masing-masing memiliki peran dan tanggung jawab yang berbeda. Setiap lapisan saling terhubung melalui protokol komunikasi yang telah ditetapkan, dengan arah aliran data yang terdefinisi dalam berbagai *layer*:



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

1. Lapisan Pengguna (User Layer)

Lapisan ini merupakan titik interaksi utama bagi pengguna sistem, baik sebagai Administrator maupun Executor. Pengguna mengakses *Unified Dashboard* melalui *web browser* standar. Aliran data pada lapisan ini bersifat dua arah (*bi-directional*) menggunakan protokol *HTTPS* (Port 443), di mana browser mengirimkan request (seperti konfigurasi sistem atau *query* investigasi) dan sistem mengembalikan respon berupa *Graphical User Interface* yang telah di-render.

2. Lapisan Aplikasi (Application Layer)

Lapisan ini bertindak sebagai "otak" sistem yang mencakup logika *backend* dan penyimpanan data. Untuk menjaga performa tetap optimal, lapisan ini dibagi menjadi dua proses komputasi utama dan dua komponen penyimpanan data:

- a. Proses Komputasi:
 - *Web Server* (Flask App): Bertanggung jawab melayani permintaan langsung dari pengguna, memvalidasi *form*, dan menangani pemanggilan API secara *on-demand*.
 - *Background Scheduler*: Bertindak sebagai *engine* otonom (*Sentinel*) yang bekerja secara asinkron di latar belakang. Komponen ini melakukan *health check* proaktif dan memproses antrean notifikasi tanpa membebani *thread* utama. Komunikasi antara *Web Server* dan *Scheduler* dijembatani secara satu arah melalui mekanisme *Publish/Subscribe* (Pub/Sub) pada Redis.
- b. Komponen Penyimpanan Data:
 - PostgreSQL (Port 5432/TCP): Berfungsi sebagai Single Source of Truth (SSOT) untuk menyimpan data konfigurasi sistem dan kredensial API. Data sensitif pada lapisan ini disimpan dengan enkripsi simetris (*Encryption at Rest*).
 - Redis (Port 6379/TCP): Berperan sebagai *in-memory* cache untuk mempercepat agregasi data dan sebagai basis data *State Machine*



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

untuk melacak status kesehatan (*uptime/downtime*) komponen secara *real-time*.

3. Lapisan Integrasi Eksternal (*External Integration Layer*)

Lapisan ini mencakup infrastruktur luar yang diorkestrasi oleh sistem. Komunikasi dilakukan melalui *Application Programming Interface* (API) dengan aturan protokol dan port yang spesifik:

- a. Wazuh Infrastructure:
 - *Wazuh Agent*: Beroperasi di level *endpoint* dan mengirimkan telemetry keamanan ke *Wazuh Manager* melalui Port 1514/TCP secara satu arah. Aplikasi Hub tidak berinteraksi langsung dengan agen.
 - *Wazuh Manager & Indexer*: Aplikasi Hub berkomunikasi secara dua arah dengan *Manager* (Port 55000/TCP) dan *Indexer* (Port 9200/TCP) untuk menarik data inventaris dan memvalidasi kesehatan kluster menggunakan autentikasi API.
- b. AlienVault OTX (*Threat Intelligence*): Interaksi bersifat dua arah melalui Port 443/TCP (*HTTPS*). *Web Server* mengirimkan *Indicator of Compromise* (IoC) dan menerima data konteks ancaman dalam format JSON.
- c. Telegram Bot API: Interaksi bersifat satu arah (*Push-only*) melalui Port 443/TCP (*HTTPS*). *Scheduler* hanya mengirimkan notifikasi peringatan dini ke *Chat ID* tujuan tanpa melakukan *polling* balasan, guna menjaga efisiensi proses latar belakang.

4.2.2 Rancangan Optimalisasi Alur Kerja Operasional (*As-Is* dan *To-Be*)

Subbab ini merinci bagaimana terjadinya transformasi alur kerja operasional di lingkungan *Security Operations Center* (SOC) PT. XYZ melalui implementasi sistem yang diusulkan. Pendekatan perancangan pada bagian ini mengadopsi pemodelan *As-Is* dan *To-Be* untuk memetakan secara presisi *area* inefisiensi yang akan direduksi.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Kondisi *As-Is* merepresentasikan keadaan eksisting operasional SOC saat ini. Kondisi ini ditandai dengan fragmentasi perangkat, tingginya beban kerja administratif manual, dan ketergantungan penuh pada antarmuka baris perintah (CLI). Dalam penelitian ini, kondisi *As-Is* diposisikan sebagai *baseline* atau garis ukur dasar dari prosedur manual sebelum adanya intervensi sistem.

Sebaliknya, kondisi *To-Be* merepresentasikan proyeksi arsitektur operasional *ideal* yang ingin dicapai, yang diwujudkan melalui sistem usulan. Sistem usulan ini dirancang berupa Hub Manajemen Terpusat yang bertindak mengabstraksikan seluruh kompleksitas teknis dari kondisi *As-Is* menjadi antarmuka visual yang terotomatisasi (*Single Pane of Glass*).

Perbandingan di bawah ini menunjukkan rancangan efisiensi langkah kerja yang dihasilkan oleh sistem usulan (*To-Be*) dibandingkan dengan prosedur manual (*As-Is*) pada sepuluh aktivitas inti SOC. Rincian optimalisasi pada masing-masing aksi diuraikan sebagai berikut:

1. Manajemen Komponen Infrastruktur (Wazuh Hub)

Bagian ini membahas implementasi Wazuh Hub dalam mendukung pengelolaan infrastruktur secara terpusat melalui beberapa aktivitas administrasi utama, yaitu: a) inventarisasi komponen baru, b) pemeliharaan kredensial (perubahan *password*), c) pemantauan kesehatan multi-komponen, dan d) inventarisasi daftar agen (*agent list*). Setiap aktivitas membandingkan alur kerja konvensional dengan sistem usulan untuk menunjukkan peningkatan efisiensi pengelolaan infrastruktur.

1a. Aktivitas Inventarisasi Komponen Baru

Aktivitas ini dilakukan saat perusahaan melakukan ekspansi infrastruktur dengan menambahkan Wazuh *Manager* atau *Indexer* baru ke dalam lingkungan operasional, yang menuntut integrasi segera agar log keamanan dapat terpantau. Tabel 4.3 membandingkan perbandingan langkah alur kerja inventarisasi komponen baru dengan asumsi proses instalasi dan *setup* telah selesai:



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Tabel 4.3. Perbandingan Alur Kerja Inventarisasi Komponen Baru

No	Metode Manual (<i>Existing</i>)	Metode Usulan (<i>Proposed</i>)
1	Administrator selesai melakukan deployment dan <i>setup</i> konfigurasi komponen Wazuh baru (<i>Manager/ Indexer</i>)	Administrator selesai melakukan deployment dan <i>setup</i> konfigurasi komponen Wazuh baru (<i>Manager/ Indexer</i>)
2	Administrator masuk ke <i>server</i> sentral via SSH.	Administrator mengakses halaman konfigurasi Hub untuk mengisi detail URL dan kredensial baru.
3	Administrator mencatat URL dan API <i>Port</i> secara manual.	Administrator mengklik simpan.
4	Administrator menyimpan kredensial ke <i>file</i> lokal.	Sistem melakukan validasi koneksi secara otomatis.
5	Administrator mengkonfirmasi akses ke komponen.	Sistem mengenkripsi dan menyimpan data ke <i>database</i> pusat.
6	Administrator melakukan verifikasi koneksi <i>end-to-end</i> .	-
Total	6 Langkah Kerja (Administrator)	3 Langkah Kerja (Administrator)

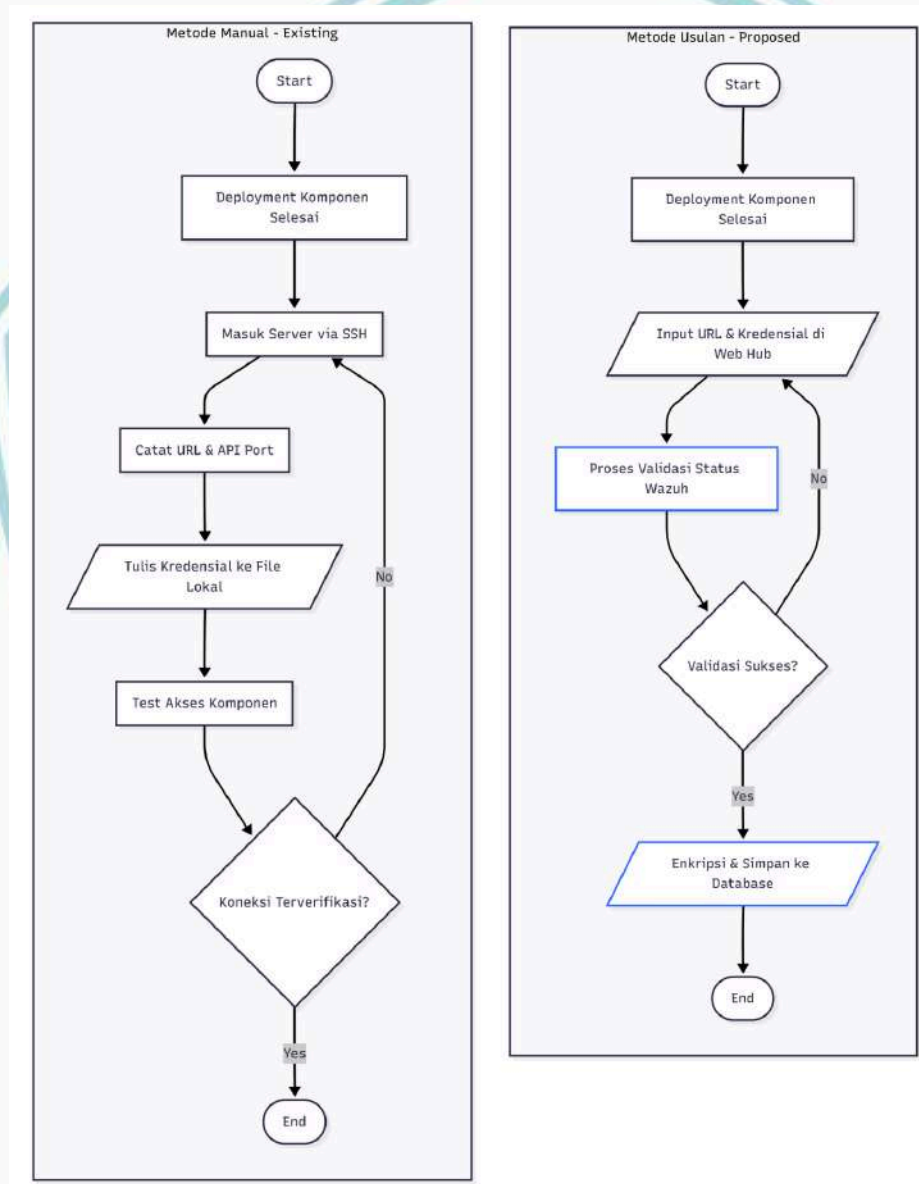
Perbedaan langkah kerja yang disajikan pada Tabel 4.3 kemudian disederhanakan ke dalam bentuk diagram alur proses agar alur kerja sistem lebih mudah dipahami. Pada diagram tersebut, proses yang dijalankan secara otomatis oleh sistem ditandai dengan garis berwarna biru.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Visualisasi pada Gambar 4.2 ini memperlihatkan secara jelas bagaimana pendekatan yang diusulkan mampu mengurangi tahapan interaksi yang sebelumnya harus dilakukan secara berulang pada level server. Dengan demikian, sistem tidak hanya menyederhanakan proses inventarisasi komponen baru, tetapi juga mengeliminasi redundansi interaksi sehingga pengelolaan infrastruktur menjadi lebih efisien, konsisten, dan mudah dilakukan.



Gambar 4.2 Diagram Alur Komparasi Aktivitas Inventarisasi Komponen Baru Metode Manual (Existing) dan Metode Usulan (Proposed)



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

1b. Aktivitas Pemeliharaan Kredensial (Perubahan *Password*)

Aktivitas ini dilakukan saat organisasi melakukan rotasi keamanan rutin atau terjadi pergantian *password* pada komponen infrastruktur secara mandiri di sisi *server*, sehingga kredensial pada pencatatan tim SOC harus diselaraskan kembali (*sync*).

Tabel 4.4. Perbandingan Alur Kerja Pemeliharaan Kredensial

No	Metode Manual (<i>Existing</i>)	Metode Usulan (<i>Proposed</i>)
1	Administrator memperbarui kredensial pada tiap komponen <i>Indexer/Manager</i> .	Administrator memperbarui kredensial pada tiap komponen <i>Indexer/Manager</i> .
2	Administrator memverifikasi apakah kredensial baru telah berfungsi dengan <i>login</i> satu-per-satu.	Administrator menekan tombol Edit pada komponen terkait di halaman menu <i>Manage Wazuh</i> .
3	Administrator mengakses <i>file</i> penyimpanan kredensial di folder lokal.	Administrator mengetikkan <i>password</i> baru ke antarmuka Hub untuk divalidasi dan disimpan sistem.
4	Administrator menghapus catatan kredensial lama dan menggantinya dengan yang baru.	Sistem melakukan <i>re-validation</i> koneksi secara instan.
Total	4 Langkah Kerja (Administrator)	3 Langkah Kerja (Administator)

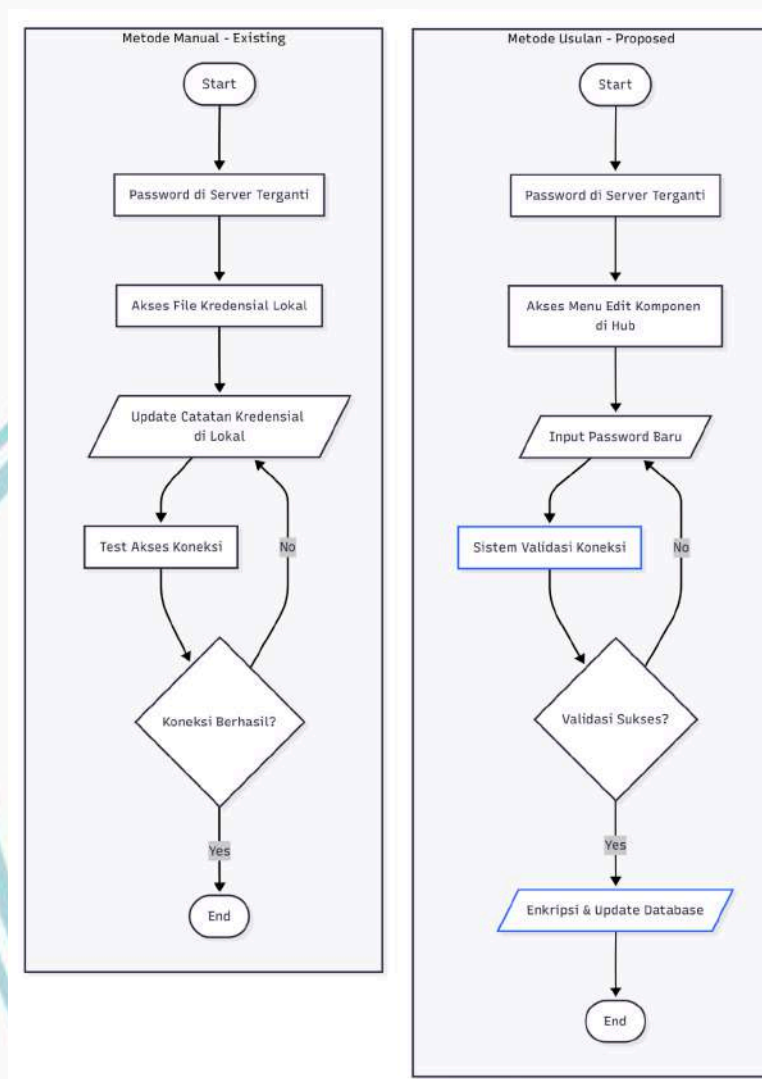
Berdasarkan komparasi pada Tabel 4.4, alur operasional dalam melakukan pembaruan kredensial dipetakan ke dalam diagram alur proses pada Gambar 4.3 (proses yang dilakukan oleh sistem ditandai dengan garis biru) untuk



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang menggunakan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

memvisualisasikan bagaimana validasi instan pada metode usulan meminimalisir risiko kesalahan pencatatan.



Gambar 4.3 Diagram Alur Komparasi Aktivitas Pemeliharaan Kredensial (Perubahan *Password*) Metode Manual (Kiri) dan Metode Usulan (Kanan)

1c. Aktivitas Pemantauan Kesehatan Multi-Komponen

Aktivitas ini dilakukan saat administrator perlu memastikan ketersediaan layanan (availability) dari seluruh kluster Wazuh (dua Wazuh *Indexer* (Port 9200) dan satu Wazuh *Manager* (Port 55000)) guna memastikan seluruh *node* dalam kondisi *online* dan siap menerima data. Tabel 4.5 menyajikan perbandingan alur kerja pemantauan kesehatan multi-komponen antara pendekatan konvensional dan



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

sistem usulan, sehingga perbedaan tahapan serta peningkatan efisiensi yang diperoleh dapat terlihat dengan lebih jelas.

Tabel 4.5. Perbandingan Alur Kerja Pemantauan Kesehatan Multi-Komponen

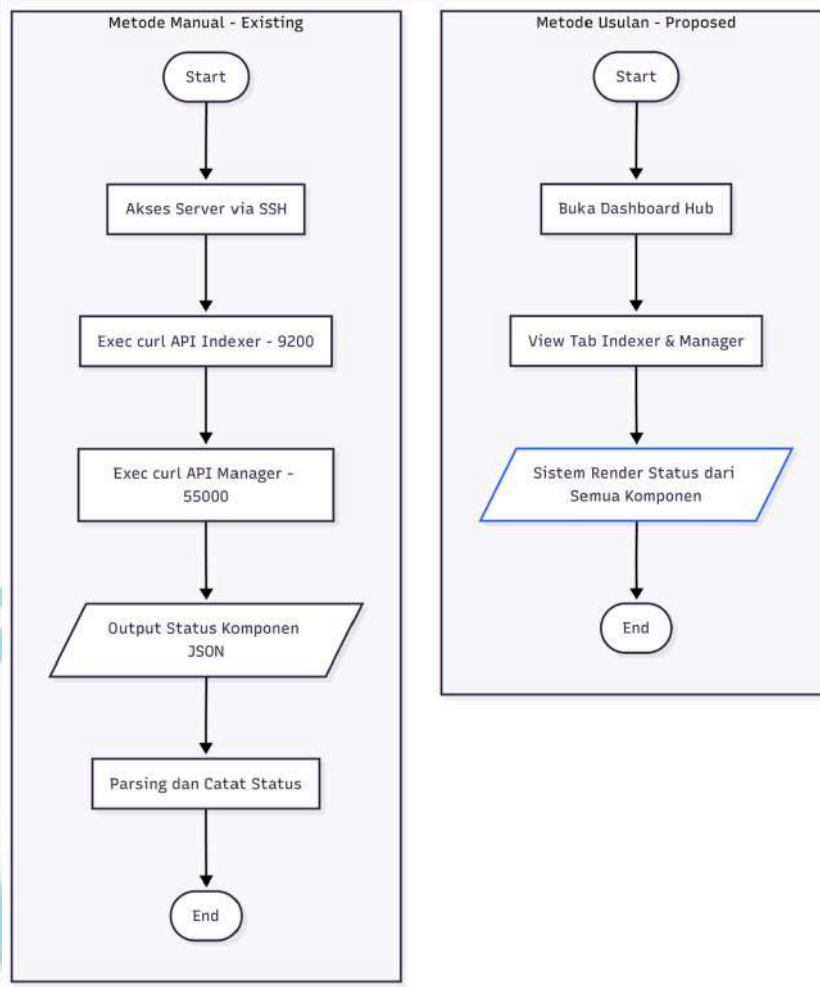
No	Metode Manual (<i>Existing</i>)	Metode Usulan (<i>Proposed</i>)
1	Administrator membuka <i>terminal</i> untuk akses <i>server</i> via SSH.	Administrator mengakses halaman <i>Dashboard</i> Hub dan mengakses tab <i>Indexer</i> .
2	Administrator menjalankan perintah <i>curl</i> ke API <i>Indexer</i> (Port 9200).	Sistem menampilkan status seluruh komponen <i>Indexer</i> secara <i>live</i> .
3	Administrator menjalankan perintah <i>curl</i> ke API <i>Manager</i> (Port 55000).	Administrator mengganti tab agar sistem menampilkan status seluruh komponen <i>Manager</i> secara <i>live</i> .
4	Administrator membaca respons JSON secara manual pada tiap <i>node</i> .	-
5	Administrator mencatat status operasional tiap <i>node</i> .	-
Total	5 Langkah Kerja (Administrator)	2 Langkah Kerja (Administrator)

Prosedur operasional dalam melakukan verifikasi status kesehatan kluster multi-*node* dijelaskan melalui diagram alur pada Gambar 4.4. Diagram ini memvisualisasikan bagaimana transisi dari pengecekan berbasis *command-line* yang bersifat manual menuju otomatisasi pemantauan berbasis *dashboard* dapat meminimalisir latensi operasional bagi tim SOC.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 4.4 Diagram Alur Komparasi Aktivitas Pemantauan Kesehatan Multi-Komponen Metode Manual (Kiri) dan Metode Usulan (Kanan)

1d. Aktivitas Inventarisasi Daftar Agen (*Agent List*)

Aktivitas ini dilakukan saat administrator perlu memverifikasi daftar seluruh agen yang terdaftar di berbagai *server* Wazuh untuk memastikan tidak ada *endpoint* yang terlewat dari pemantauan keamanan. Tabel 4.6 menyajikan perbandingan alur kerja inventarisasi daftar agen antara pendekatan konvensional dan sistem usulan, sehingga perbedaan tahapan serta peningkatan efisiensi yang dihasilkan dapat diamati dengan lebih jelas.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Tabel 4.6. Perbandingan Alur Kerja Inventarisasi Daftar Agen

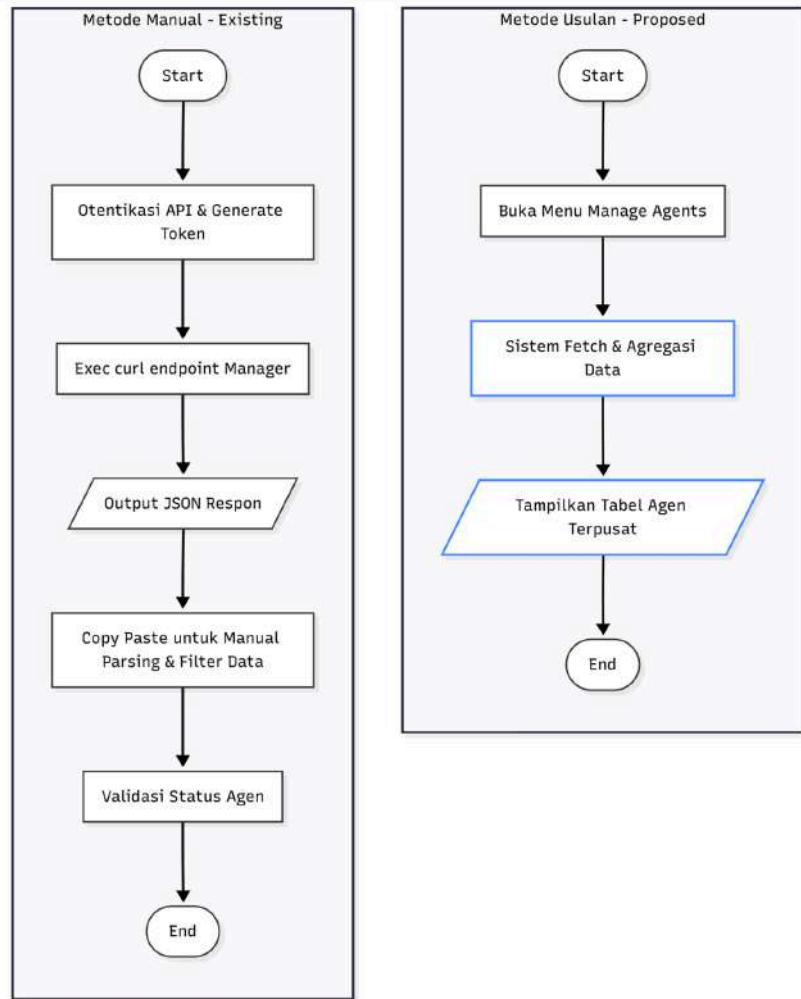
No	Metode Manual (<i>Existing</i>)	Metode Usulan (<i>Proposed</i>)
1	Administrator melakukan otentikasi API untuk mendapatkan <i>token</i> akses.	Administrator mengakses menu <i>Manage Agents</i> pada Hub.
2	Administrator menjalankan perintah curl ke <i>endpoint</i> API <i>Manager</i> .	Sistem secara otomatis menarik seluruh daftar agen dari semua <i>Manager</i> .
3	Administrator menyalin hasil respons JSON yang berisi daftar status.	Sistem menampilkan daftar agen dalam tabel teragregasi.
4	Administrator melakukan pembersihan data untuk membaca daftar agen.	-
5	Administrator memvalidasi status setiap agen secara visual.	-
Total	5 Langkah Kerja (Administrator)	1 Langkah Kerja (Administrator)

Berdasarkan komparasi pada Tabel 4.6, alur kerja inventarisasi *endpoint* agen selanjutnya dipetakan ke dalam diagram alur pada Gambar 4.5 untuk memberikan gambaran visual mengenai perbedaan tahapan antara metode konvensional dan sistem usulan. Diagram tersebut menunjukkan bagaimana sistem usulan mengeliminasi kebutuhan proses *parsing* data secara manual serta mengurangi proses autentikasi API yang harus dilakukan berulang kali pada setiap server. Penyederhanaan alur ini membuat proses inventarisasi daftar agen menjadi lebih terstruktur, mengurangi interaksi administratif yang tidak diperlukan, serta memungkinkan informasi agen dari seluruh server diperoleh melalui mekanisme yang lebih efisien dan terpusat.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 4.5 Diagram Alur Komparasi Aktivitas Inventarisasi Daftar Agen (*Agent List*) Metode Manual (Kiri) dan Metode Usulan (Kanan)

2. Pengayaan *Threat Intelligence*

Bagian ini membahas implementasi fitur pengayaan *threat intelligence* untuk mendukung proses analisis dan investigasi keamanan secara lebih cepat dan terpusat. Aktivitas yang dibahas meliputi: a) investigasi *on-demand*, b) investigasi *alert (click-to-investigate)*, dan c) konfigurasi API *threat intelligence*. Setiap aktivitas membandingkan alur kerja konvensional dengan sistem usulan untuk menunjukkan bagaimana integrasi *threat intelligence* mampu menyederhanakan proses investigasi, mempercepat akses terhadap informasi ancaman yang relevan, serta membuat analisis insiden keamanan yang lebih efektif.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

2a. Aktivitas Investigasi *On-Demand*

Aktivitas ini dilakukan saat analis menerima detail IP atau *hash* ancaman melalui *ticket* atau request langsung dari pihak eksternal/internal yang perlu diselidiki status reputasinya secara mendalam.

Tabel 4.7. Perbandingan Alur Kerja Investigasi *On-Demand*

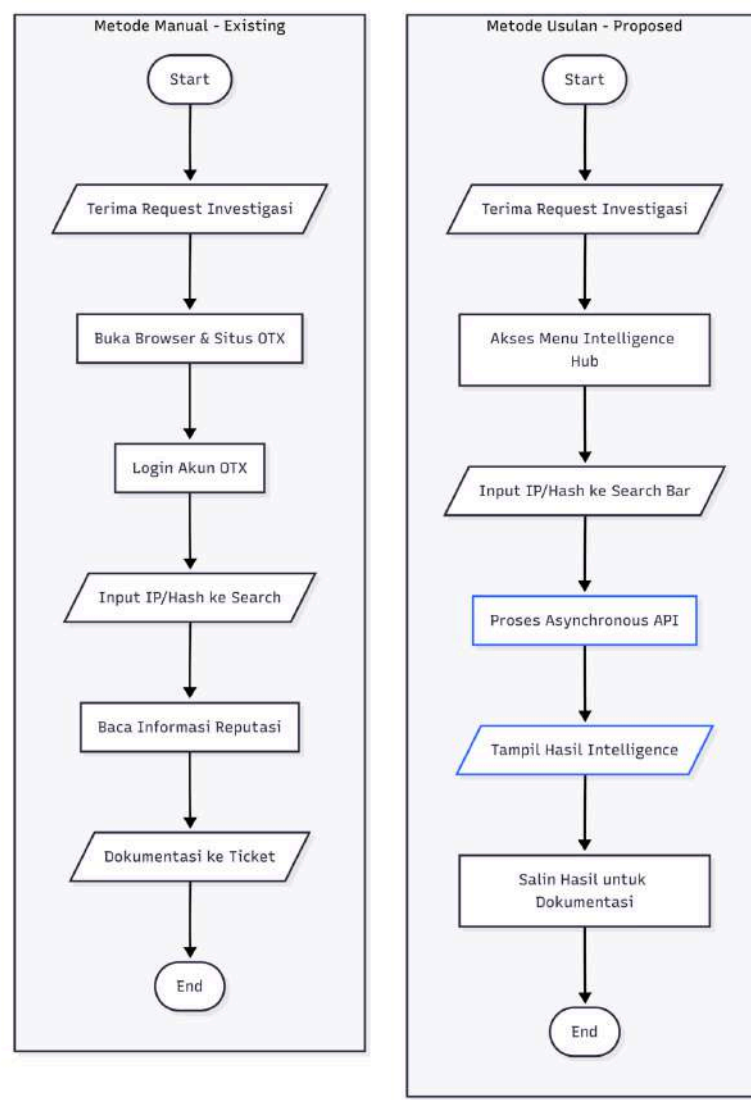
No	Metode Manual (<i>Existing</i>)	Metode Usulan (<i>Proposed</i>)
1	Analisis menerima request untuk investigasi sebuah IP/ <i>Hash</i>	Analisis menerima request untuk investigasi sebuah IP/ <i>Hash</i>
2	Analisis membuka <i>browser</i> dan menuju situs AlienVault OTX.	Analisis mengakses menu <i>Intelligence</i> pada Hub.
3	Analisis melakukan <i>login</i> ke akun OTX.	Analisis memasukkan IP/ <i>Hash</i> ke kolom pencarian intelijen.
4	Analisis menyalin IP/ <i>Hash</i> dari <i>ticket/request</i> dan melakukan <i>paste</i> di kolom pencarian.	Sistem memproses kueri API secara <i>asynchronous</i> .
5	Analisis menyalin satu per-satu informasi reputasi, <i>threat pulse</i> , dan geolokasi.	Sistem menampilkan ringkasan intelijen, <i>threat pulse</i> , dan detail forensik.
6	Analisis mendokumentasikan hasil temuan ke dalam dokumen.	Analisis menyalin hasil temuan untuk didokumentasikan.
Total	6 Langkah Kerja (Analisis)	4 Langkah Kerja (Analisis)

Perbandingan langkah kerja pada Tabel 4.7 ditransformasikan ke dalam diagram alur pada Gambar 4.6 di bawah ini, yang menyoroti bagaimana integrasi API secara *asynchronous* pada Hub Manajemen mampu menghilangkan hambatan context-switching saat melakukan investigasi mandiri.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 4.6 Diagram Alur Komparasi Aktivitas Investigasi *On-Demand* Metode Manual (Kiri) dan Metode Usulan (Kanan)

2b. Aktivitas Investigasi *Alert (Click-to-Investigate)*

Aktivitas ini dilakukan saat analis menemukan sebuah indikator IP dari log Wazuh yang reputasinya perlu segera diverifikasi untuk menentukan apakah IP tersebut merupakan ancaman nyata atau *false positive*. Proses ini penting karena informasi reputasi IP menjadi salah satu dasar dalam menilai tingkat risiko suatu aktivitas, sehingga analis dapat menentukan langkah investigasi atau respons yang paling sesuai secara lebih cepat dan akurat.



Tabel 4.8. Perbandingan Alur Kerja Investigasi *Alert (Click-to-Investigate)*

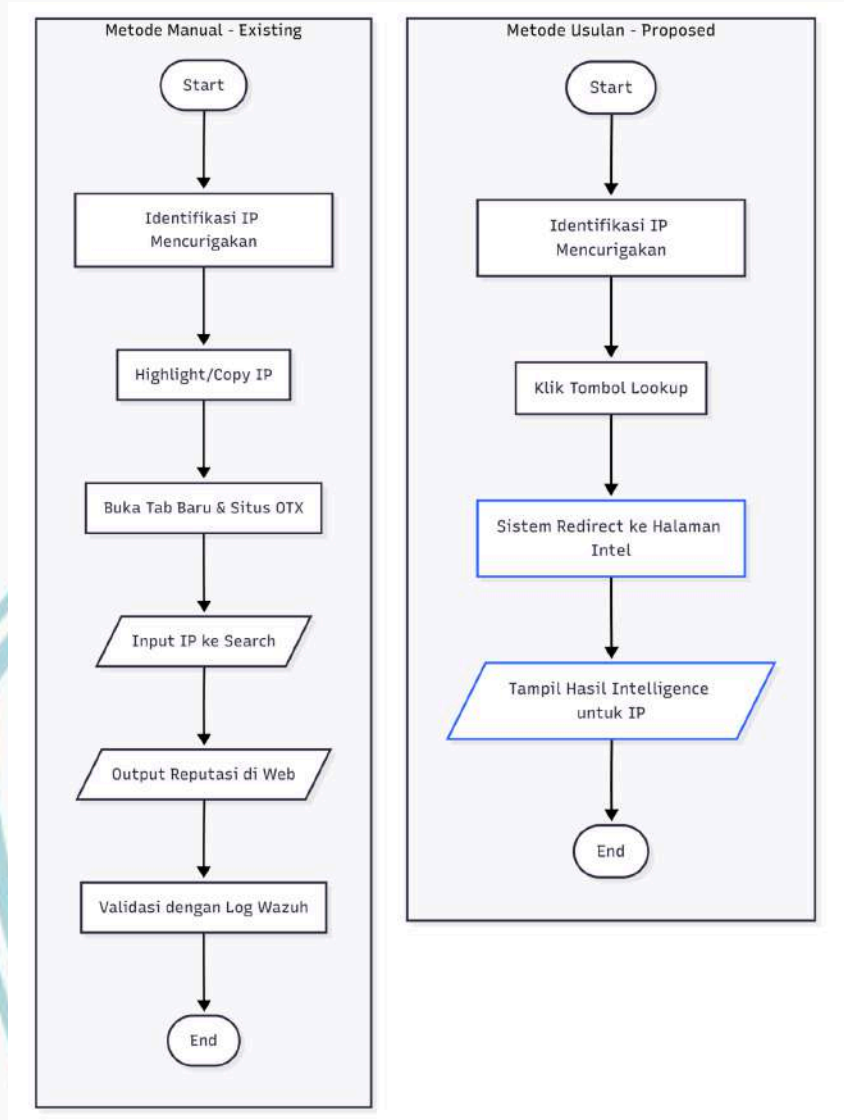
No	Metode Manual (<i>Existing</i>)	Metode Usulan (<i>Proposed</i>)
1	Analisis mengidentifikasi IP mencurigakan pada daftar <i>alert</i> .	Analisis mengidentifikasi IP mencurigakan pada daftar <i>alert</i> .
2	Analisis melakukan <i>highlight</i> (sorot) pada teks IP di log tersebut.	Analisis mengklik tombol <i>Lookup</i> di samping alamat IP.
3	Analisis membuka <i>tab browser</i> baru dan menuju situs OTX.	Sistem secara otomatis mengarahkan ke halaman intelijen.
4	Analisis melakukan <i>paste</i> IP ke kolom pencarian OTX.	Sistem menampilkan hasil analisis intelijen secara instan.
5	Analisis memvalidasi hasil reputasi dengan <i>alert</i> asli di log Wazuh.	-
Tot al	5 Langkah Kerja (Analisis)	2 Langkah Kerja (Analisis)

Alur investigasi *Click-to-Investigate* pada Gambar 4.7 memvisualisasikan reduksi langkah kerja yang signifikan, dari lima tahapan pada metode konvensional menjadi hanya dua tahapan pada sistem usulan. Penyederhanaan ini dimungkinkan oleh integrasi langsung antara antarmuka dashboard Hub dan layanan *Threat Intelligence*, sehingga proses investigasi dapat dilakukan tanpa perlu melakukan *copy-paste* indikator maupun berpindah antar tab atau aplikasi. Dengan demikian, analisis dapat mengakses informasi reputasi ancaman secara lebih cepat melalui satu alur kerja yang terintegrasi, sehingga proses investigasi menjadi lebih efisien dan minim langkah manual.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 4.7 Diagram Alur Komparasi Aktivitas Investigasi *Alert* (*Click-to-Investigate*) Metode Manual (Kiri) dan Metode Usulan (Kanan)

2c. Aktivitas Konfigurasi API *Threat Intelligence*

Aktivitas ini dilakukan saat sistem pertama kali diimplementasikan atau ketika dilakukan pembaruan *API Key* secara berkala guna memastikan integrasi dengan layanan *Threat Intelligence* tetap aktif, terotorisasi, dan dapat digunakan tanpa kendala. Konfigurasi yang benar diperlukan agar proses pengambilan informasi ancaman dapat berlangsung konsisten dan mendukung aktivitas investigasi keamanan.



Tabel 4.9. Perbandingan Alur Kerja Konfigurasi API *Threat Intelligence*

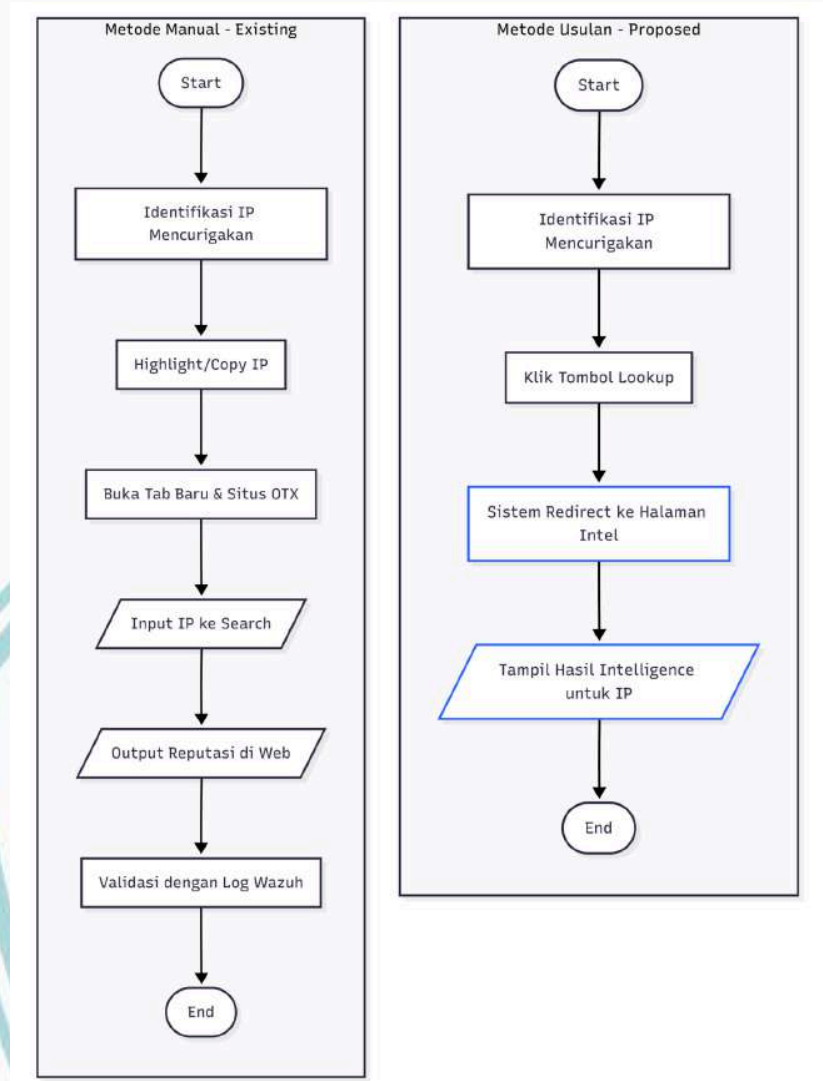
No	Metode Manual (<i>Existing</i>)	Metode Usulan (<i>Proposed</i>)
1	Administrator telah membuat akun Alienvault OTX dan membuat API <i>key</i> .	Administrator telah membuat akun Alienvault OTX dan membuat API <i>key</i> .
2	Administrator membuka situs AlienVault untuk membuka menu kredensial API.	Administrator mengakses menu API <i>Configuration</i> pada Hub.
3	Administrator menyalin API <i>Key</i> ke <i>file</i> konfigurasi sistem secara manual.	Administrator memasukkan API <i>Key</i> ke formulir konfigurasi Hub untuk disimpan.
4	Administrator melakukan verifikasi integrasi via terminal/CLI.	Sistem mengenkripsi dan menyimpan kredensial API secara otomatis.
Tot al	4 Langkah Kerja (Administrator)	3 Langkah Kerja (Administrator)

Diagram alur pada Gambar 4.8 memperlihatkan bagaimana proses konfigurasi kredensial pada metode usulan dilakukan dengan alur yang lebih aman, terstruktur, dan sederhana dibandingkan metode manual. Perbedaan tersebut dicapai melalui mekanisme enkripsi otomatis yang diterapkan sistem saat proses penyimpanan data kredensial, sehingga administrator tidak perlu lagi melakukan pengelolaan enkripsi secara manual. Pendekatan ini di samping menyederhanakan tahapan konfigurasi juga membantu meningkatkan keamanan penyimpanan kredensial serta mengurangi risiko kesalahan konfigurasi yang dapat memengaruhi integritas sistem.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 4.8 Diagram Alur Komparasi Aktivitas Konfigurasi API *Threat Intelligence* Metode Manual (Kiri) dan Metode Usulan (Kanan)

3. Notifikasi & *Engine* Pemantauan Sistem

Bagian ini membahas implementasi fitur notifikasi dan *engine* pemantauan sistem untuk memastikan mekanisme peringatan dapat dikonfigurasi dan beroperasi dengan baik. Aktivitas yang dibahas meliputi: a) konfigurasi notifikasi dan b) uji coba integrasi (*connectivity test*). Setiap aktivitas membandingkan alur kerja konvensional dengan sistem usulan untuk menunjukkan penyederhanaan proses konfigurasi serta peningkatan kemudahan dalam memvalidasi mekanisme pengiriman notifikasi.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

3a. Aktivitas Konfigurasi Notifikasi

Aktivitas ini dilakukan saat sistem baru diimplementasikan atau saat terjadi perubahan kebijakan pada alur notifikasi, di mana administrator perlu melakukan pengaturan koneksi antara sistem Hub dengan platform pesan instan (Telegram Bot) untuk menyampaikan peringatan sistem.

Tabel 4.10. Perbandingan Alur Kerja Konfigurasi Notifikasi Telegram

No	Metode Manual (<i>Existing</i>)	Metode Usulan (<i>Proposed</i>)
1	Administrator membuka terminal/SSH ke <i>server</i> target.	Administrator mengakses menu <i>Notifications</i> pada Hub.
2	Administrator mencari dan mengedit <i>file</i> skrip notifikasi manual.	Administrator memasukkan Bot <i>Token</i> dan <i>Chat ID</i> ke formulir.
3	Administrator menyalin <i>token</i> dan <i>ID</i> ke dalam <i>file</i> sistem.	Administrator menekan tombol simpan.
4	Administrator mengatur <i>permission</i> (<i>chmod/chown</i>) skrip.	-
5	Administrator mengonfigurasi <i>ossec.conf</i> untuk integrasi.	-
6	Administrator melakukan <i>restart</i> layanan Wazuh.	-
Tot al	6 Langkah Kerja (Administrator)	3 Langkah Kerja (Administrator)

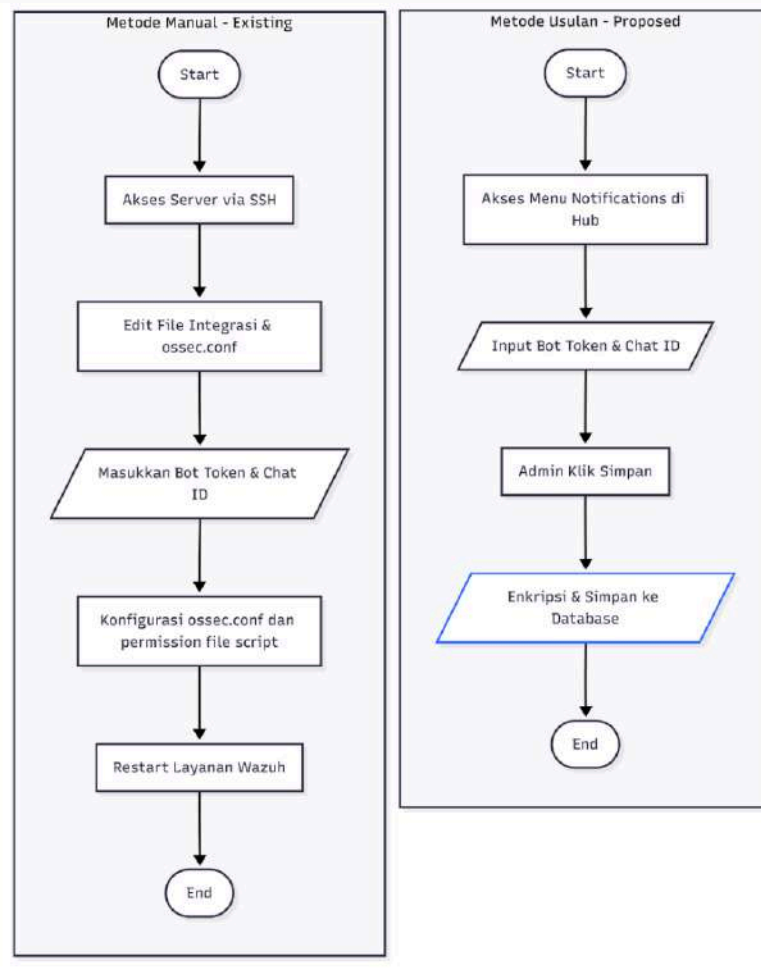
Proses konfigurasi notifikasi yang sebelumnya melibatkan interaksi tingkat sistem pada *file* konfigurasi (*ossec.conf*) dan skrip integrasi, kini disederhanakan melalui antarmuka terpusat pada sistem Hub. Diagram alur pada



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Gambar 4.9 di bawah ini memvisualisasikan transisi dari alur kerja berbasis *command-line* yang rentan terhadap *human error* menuju sistem konfigurasi berbasis web yang lebih aman dan efisien.



Gambar 4.9 Diagram Alur Komparasi Aktivitas Konfigurasi Notifikasi Metode Manual (Kiri) dan Metode Usulan (Kanan)

3b. Aktivitas Uji Coba Integrasi (Connectivity Test)

Aktivitas ini dilakukan untuk memvalidasi bahwa kredensial Telegram Bot yang telah dikonfigurasi sudah benar dan memiliki hak akses yang diperlukan untuk mengirimkan pesan ke *group* atau *channel* tujuan sebelum sistem dioperasikan secara penuh. Validasi ini bertujuan memastikan mekanisme notifikasi dapat berjalan dengan baik sejak awal sehingga setiap peringatan



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

keamanan dapat diterima oleh administrator secara tepat waktu tanpa mengalami kendala autentikasi maupun konfigurasi.

Tabel 4.11. Perbandingan Alur Kerja Uji Coba Integrasi (Connectivity Test)

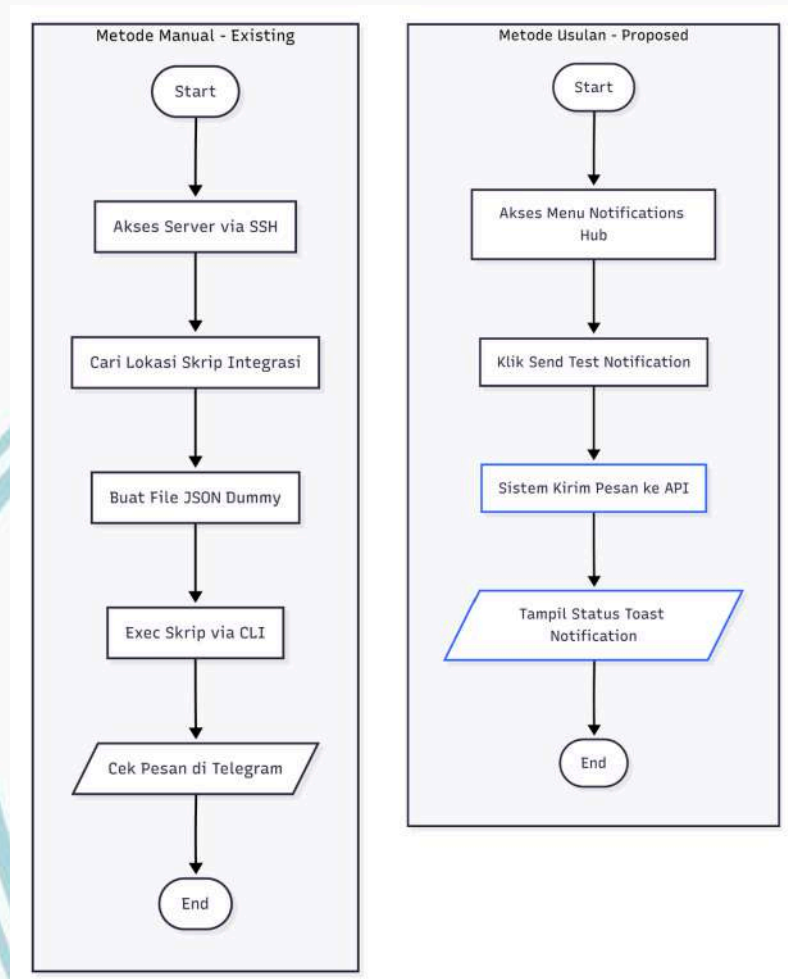
No	Metode Manual (<i>Existing</i>)	Metode Usulan (<i>Proposed</i>)
1	Administrator masuk ke <i>server</i> via SSH/terminal.	Administrator mengakses menu <i>Notifications</i> pada Hub.
2	Administrator mencari lokasi skrip integrasi pada <i>file system</i> .	Administrator menekan tombol <i>Send Test Notification</i> .
3	Administrator membuat <i>file JSON dummy</i> untuk data uji.	Sistem mengirimkan pesan tes ke API Telegram.
4	Administrator mengeksekusi skrip via baris perintah (<i>CLI</i>).	Sistem menampilkan status keberhasilan (<i>Toast Notification</i>).
5	Administrator mengecek aplikasi Telegram untuk melihat hasil.	-
Tot al	5 Langkah Kerja (Administrator)	2 Langkah Kerja (Administrator)

Untuk memvalidasi apakah integrasi Telegram berfungsi dengan baik, sistem lama mengharuskan administrator menjalankan serangkaian *command* teknis pada level sistem operasi untuk menguji proses pengiriman notifikasi. Pendekatan ini memerlukan pengetahuan teknis tambahan serta melibatkan beberapa tahapan yang harus dilakukan secara manual.

Sebagai perbandingan, diagram alur pada Gambar 4.10 menyajikan perbedaan kompleksitas antara pendekatan tersebut dengan metode usulan, yang telah mengabstraksi seluruh proses teknis ke dalam satu tombol pengujian pada dashboard notifikasi. Dengan pendekatan usulan proses validasi integrasi menjadi



lebih sederhana, cepat, dan mudah dilakukan tanpa harus berinteraksi langsung dengan sistem operasi.



Gambar 4.10 Diagram Alur Komparasi Aktivitas Uji Coba Integrasi (Connectivity Test) Metode Manual (Kiri) dan Metode Usulan (Kanan)

3c. Aktivitas Pemantauan Status Komponen

Aktivitas ini dilakukan secara terus-menerus oleh sistem untuk mendeteksi ketersediaan (*availability*) setiap node Wazuh yang terdaftar. Pemantauan dilakukan secara berkala guna memastikan seluruh komponen tetap berada dalam kondisi *online* dan dapat menjalankan fungsinya dengan baik. Proses ini sangat penting untuk mencegah terjadinya kegagalan sistem secara senyap (*silent failure*), yaitu kondisi ketika suatu layanan berhenti beroperasi tanpa segera



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

diketahui oleh administrator, yang pada akhirnya dapat menyebabkan hilangnya visibilitas keamanan dan menghambat deteksi insiden saat terjadi serangan.

Tabel 4.12. Perbandingan Alur Kerja Identifikasi Insiden Down

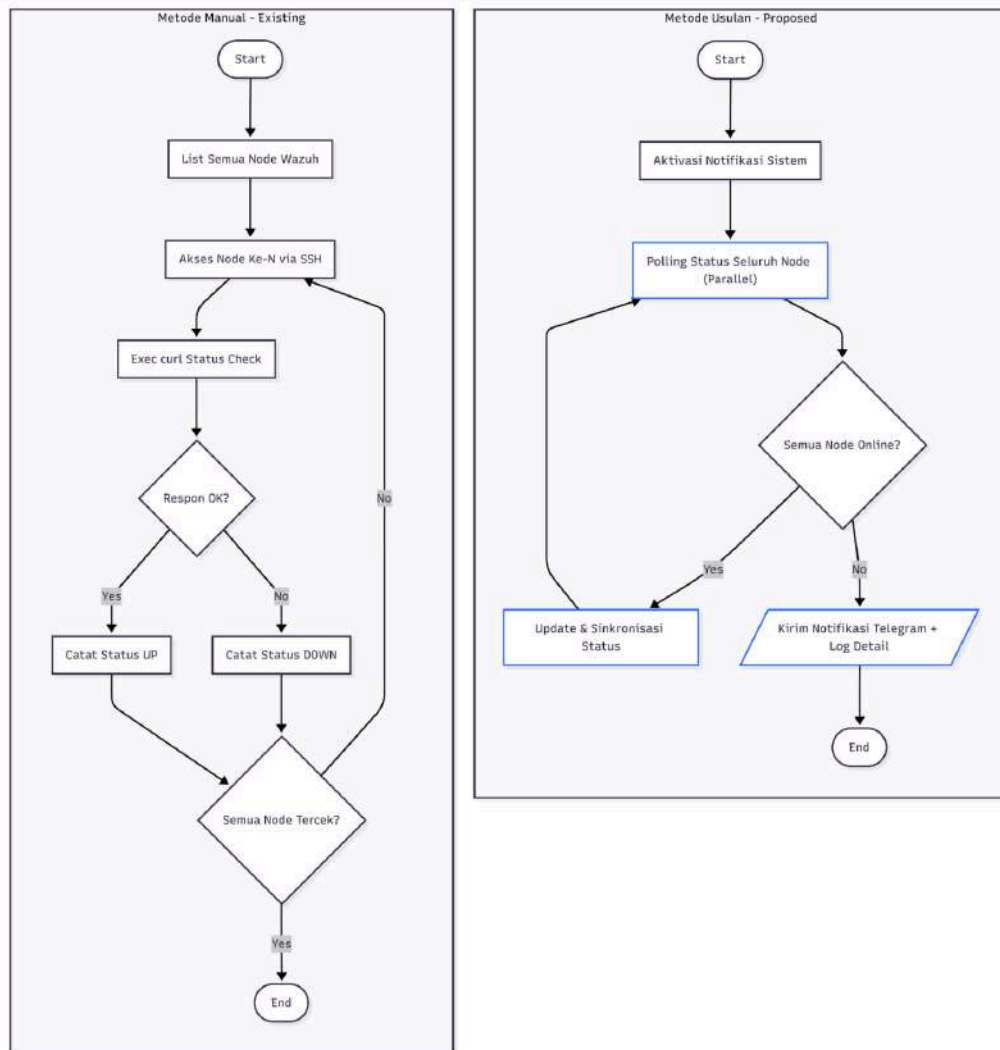
No	Metode Manual (<i>Existing</i>)	Metode Usulan (<i>Proposed</i>)
1	Administrator tidak mengetahui status terkini setiap <i>node</i> Wazuh secara <i>real-time</i> .	Administrator mengklik <i>button</i> untuk menyalakan sistem <i>notification</i> .
2	Administrator melakukan pengecekan satu per satu pada tiap <i>server</i> (manual <i>login</i>).	Sistem <i>Sentinel</i> menjalankan pemantauan <i>health check</i> secara berkala.
3	Administrator harus menunggu hingga ada laporan insiden untuk mengetahui <i>node down</i> .	Sistem melakukan validasi koneksi ke semua <i>node</i> secara paralel untuk deteksi apabila ditemukan status <i>offline</i> .
4	Administrator mendiagnosis penyebab kegagalan secara manual via terminal.	Sistem memberikan notifikasi <i>real-time</i> ke Telegram beserta detail kegagalan.
Total	4 Langkah Kerja (Administrator)	1 Langkah Kerja (Administrator)

Transformasi alur kerja dari pemantauan manual yang bergantung pada pelaporan insiden menuju sistem pemantauan proaktif divisualisasikan pada Gambar 4.11. Diagram ini memperlihatkan bagaimana sistem *Sentinel* berperan sebagai pemantau status tiap komponen, sehingga mengurangi *blind spot* yang sebelumnya dialami oleh tim SOC.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 4.11 Diagram Alur Komparasi Aktivitas Pemantauan Status Komponen Metode Manual (Kiri) dan Metode Usulan (Kanan)

4.2.3 Perancangan Penyimpanan Data (*Cache-Database Architecture*)

Sistem ini dirancang untuk menangani beban kerja yang sangat kontras: di satu sisi, sistem membutuhkan akses instan ke data operasional yang terus berubah, namun di sisi lain, sistem wajib menjaga keamanan absolut terhadap data kredensial yang permanen. Untuk mengakomodasi dua kebutuhan ini, sistem memanfaatkan paradigma *Polyglot Persistence*, yaitu penggunaan teknologi basis data yang berbeda secara bersamaan berdasarkan karakteristik siklus hidup data. Dalam implementasinya, literatur mengidentifikasi bentuk terapannya sebagai arsitektur *cache-database*.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Arsitektur penyimpanan ini membagi beban kerja ke dalam dua lapisan utama: lapisan volatil berkinerja tinggi menggunakan Redis, dan lapisan persisten yang aman menggunakan PostgreSQL. Pendekatan ini memastikan antarmuka pengguna tetap responsif, di samping menjaga integritas dan keamanan konfigurasi sistem. Penjelasan implementasinya adalah sebagai berikut:

1. Lapisan Penyimpanan Volatil (Redis)

Redis secara umum memiliki peran sebagai tempat penyimpanan sementara. Dalam sistem ini, peran tersebut diperluas lagi untuk mempercepat operasi yang sifatnya asinkron di seluruh modul aplikasi. Karena beroperasi sepenuhnya di dalam memori (*in-memory*), Redis mengeliminasi latensi yang biasanya terjadi pada operasi baca-tulis perangkat keras (disk I/O). Dalam mencapai tugasnya dalam sistem ini, Redis mengemban tiga peran yang krusial yang dijelaskan dalam tabel di bawah beserta beban kerja sistem yang mendasarinya:

Tabel 4.13. Penjelasan Peran Redis dalam Sistem

No	Peran Redis	Permasalahan Sistem yang Mendasari	Penggunaan Redis dalam Sistem
1	Sebagai Lapisan <i>Caching</i>	Mengambil data langsung dari API eksternal secara berulang akan menyebabkan antarmuka pengguna <i>freeze</i> .	Sistem menggunakan Redis untuk menyimpan sementara agregasi daftar ratusan agen Wazuh, status koneksi <i>node</i> , serta hasil analisis <i>Indicator of Compromise</i> dari <i>AlienVault OTX</i> . Dengan demikian, ketika analisis memuat ulang <i>dashboard</i> ,



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

			data disajikan secara instan dari memori lokal.
2	Sebagai <i>State Machine</i>	Pada modul pemantauan status komponen (<i>health check</i>), sistem harus mampu membedakan apakah sebuah komponen Wazuh baru saja terputus atau sudah mati sejak lama.	Dibandingkan membebani PostgreSQL dengan operasi pembaruan (<i>update</i>) setiap 60 detik, memori Redis digunakan untuk melacak riwayat <i>uptime</i> dan <i>downtime</i> komponen. Hal ini secara efektif mencegah terjadinya <i>spam</i> notifikasi yang berlebihan ke Telegram (<i>alert fatigue</i>).
3	Sebagai <i>Message Broker</i>	Pengiriman pesan ke API Telegram sering kali memakan waktu beberapa detik.	Untuk mencegah antarmuka web tertahan saat analis melakukan suatu aksi, sistem memanfaatkan fitur <i>Publish/Subscribe</i> (Pub/Sub) pada Redis. Aplikasi web hanya perlu <i>publish</i> perintah ke sebuah <i>channel</i> di Redis, dan pekerja latar belakang (<i>background scheduler</i>) akan menangkap pesannya lalu mengirimkannya ke Telegram secara independen.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Untuk mencegah kebocoran memori (*memory leak*) akibat penumpukan data volatil, setiap kunci (*key*) di dalam Redis dikonfigurasi dengan batas waktu kadaluarsa (Time-to-Live / TTL) yang spesifik berdasarkan jenis modulnya, sebagaimana dijabarkan pada Tabel 4.14 berikut:

Tabel 4.14. Penjelasan Spesifikasi *Key* Redis per Modul

Pola Kunci (<i>Key Pattern</i>) / <i>Channel</i>	Modul Terkait	Fungsi dan Deskripsi	Batas Waktu (<i>TTL</i>)
<i>app:manager:{id}:token</i>	Hub Multi-Wazuh	Menyimpan sementara JWT dari API Wazuh <i>Manager</i> untuk menghindari autentikasi berulang kali.	14 Menit
<i>app:wazuh:agents_data</i>	Hub Multi-Wazuh	Menyimpan hasil agregasi daftar seluruh agen dari berbagai <i>server</i> Wazuh.	2 Menit
<i>app:intel:result:{id}</i>	Threat Intel	Menyimpan hasil <i>parsing</i> respons API <i>AlienVault OTX</i> setelah proses penarikan data.	5 Menit
<i>app:sentinel:indexer:{id}</i>	Pemantauan (<i>Sentinel</i>)	Berfungsi sebagai <i>State Machine</i> untuk melacak status <i>UP/DOWN</i> komponen dan waktu peringatan terakhir.	7 Hari



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

<i>app:notifications</i>	Notifikasi Telegram	<i>Channel</i> Pub/Sub untuk mendistribusikan antrean perintah pengiriman pesan peringatan secara <i>real-time</i> .	N/A (<i>Stream</i>)
--------------------------	---------------------	--	--------------------------

2. Lapisan Penyimpanan Persisten (PostgreSQL)

Berbanding terbalik dengan karakteristik Redis yang volatil, PostgreSQL dirancang untuk mengambil peran sebagai *Single Source of Truth* (SSOT) yang bersifat permanen. Basis data relasional ini dikhususkan untuk melakukan transaksi berbasis OLTP agar menjaga struktur konfigurasi inti operasional dan tata kelola integrasi sistem.

Fokus utama perancangan pada lapisan ini adalah penerapan prinsip *Security by Design*. Karena sistem ini menyimpan akses ke seluruh infrastruktur keamanan organisasi, perlindungan terhadap kredensial berbagai komponen tersebut sifatnya mutlak. Sistem dirancang untuk memiliki mekanisme *Encryption at Rest* menggunakan algoritma enkripsi simetris yang kuat di tingkat aplikasi.

Setiap data sensitif yang dimasukkan oleh Administrator tidak akan pernah mendarat di media penyimpanan dalam format teks *plaintext*. Sebelum disimpan ke basis data, data tersebut terlebih dahulu dienkripsi sehingga hanya dapat diakses oleh sistem melalui mekanisme yang telah ditentukan. Proses dekripsi dilakukan secara eksklusif di dalam memori (RAM) ketika sistem perlu melakukan "berjabat tangan" (*handshake*) atau autentikasi dengan API eksternal yang bersangkutan, sehingga paparan data sensitif selama proses operasional dapat diminimalkan.

Pemisahan secara logis antara data yang wajib dienkripsi dan data konfigurasi operasional yang bersifat publik (*plaintext*) di dalam skema PostgreSQL ditunjukkan pada Tabel 4.15 berikut:



Tabel 4.15. Penjelasan Spesifikasi Key Redis per Modul

1. Data Kredensial Terenkripsi (<i>Ciphertext</i>)		
Kategori Modul	Objek Data yang Disimpan	Perlakuan Keamanan
Infrastruktur Wazuh	<i>Password API Wazuh Indexer; Username & Password API Wazuh Manager.</i>	Dienkripsi secara seketika (<i>on-the-fly</i>) menggunakan kunci <i>master</i> (AES) sebelum disimpan.
Intelijen Ancaman	API Key rahasia untuk layanan <i>AlienVault</i> OTX.	Dienkripsi dengan kunci <i>master</i> aplikasi.
Notifikasi Telegram	Bot Token untuk autentikasi pengiriman pesan via API Telegram.	Dienkripsi dengan kunci <i>master</i> aplikasi.
2. Data Operasional Publik (<i>Plaintext</i>)		
Infrastruktur Wazuh	Nama tampilan (<i>Display Name</i>), Alamat URL <i>Server</i> , Port API.	Disimpan dalam format teks standar untuk efisiensi <i>query</i> .
Notifikasi Telegram	Daftar identitas percakapan tujuan (<i>Chat IDs</i>), Status <i>Toggle</i> (Aktif/Nonaktif).	Disimpan dalam format teks standar karena tidak memiliki sifat <i>executable</i> .

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

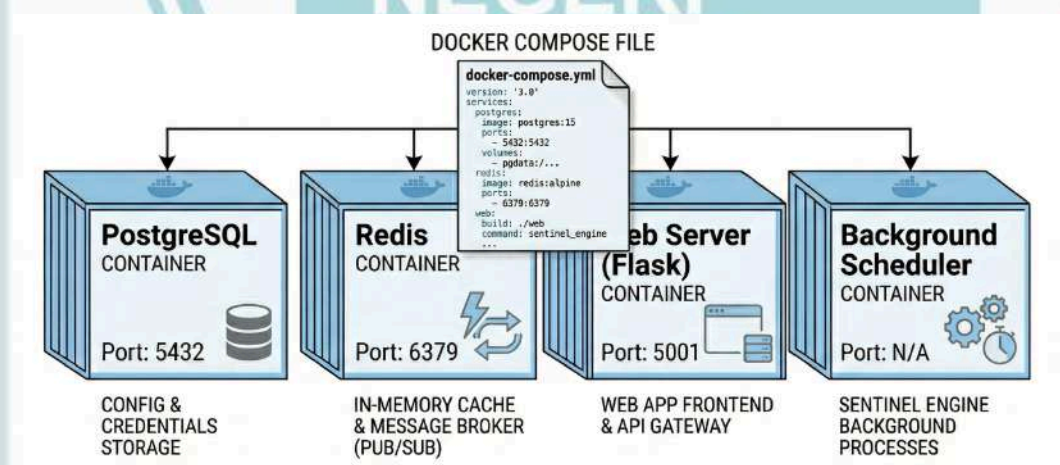


4.3 Implementasi Sistem

Subbab ini menguraikan realisasi teknis dari rancangan arsitektur menjadi sistem yang fungsional. Implementasi Hub Manajemen Keamanan Terpusat mengintegrasikan kerangka kerja *backend* Python (Flask), lapisan penyimpanan *cache-database* (Redis dan PostgreSQL), serta antarmuka web interaktif. Penjabaran teknis disusun berdasarkan urutan fungsionalitas sistem, meliputi manajemen konfigurasi infrastruktur Wazuh, komputasi paralel untuk agregasi agen, pengayaan *threat intelligence*, dan otomatisasi notifikasi. Mengingat seluruh fungsionalitas tersebut membutuhkan fondasi eksekusi yang konsisten dan terisolasi agar dapat berjalan optimal, tahapan implementasi ini diawali dengan perancangan lingkungan operasional menggunakan orkestrasi kontainer.

4.3.1 Implementasi Lingkungan Operasional (Orkestrasi Kontainer)

Implementasi sistem dilakukan menggunakan platform kontainerisasi Docker yang diorkestrasi oleh Docker Compose. Pendekatan ini dipilih untuk mencapai lingkungan eksekusi yang konsisten, terisolasi, dan mudah *dideploy* pada infrastruktur *server* target. Dengan menggunakan kontainer, setiap komponen aplikasi memiliki dependensi perangkat lunak yang mandiri, sehingga meminimalisir risiko konflik konfigurasi sistem serta mempercepat waktu pemulihan jika terjadi kegagalan pada *node server*.



Gambar 4.12 Diagram Ilustrasi Orkestrasi Kontainer dalam Docker



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Seluruh *stack* aplikasi, mulai dari lapisan basis data hingga *web server*, didefinisikan dalam konfigurasi *Docker Compose* yang terpadu untuk menjalankan 4 kontainer seperti ilustrasi pada gambar 4.12 di atas. Tabel 4.16 berikut merangkum detail spesifikasi teknis dan prasyarat operasional dari setiap layanan kontainer yang diimplementasikan:

Tabel 4.16 Spesifikasi Kontainer dan Alokasi Layanan

Nama Layanan	Fungsi Utama	Port Utama	Persyaratan/Prasyarat
PostgreSQL	<i>Relational Database</i> (Penyimpanan konfigurasi & kredensial)	5432	<i>Healthcheck</i> aktif untuk memastikan DB siap sebelum <i>App</i> berjalan.
Redis	<i>In-memory Cache & Message Broker</i> (Pub/Sub)	6379	Penyimpanan <i>volatile</i> untuk <i>state machine</i> sistem.
Web Server (Flask)	<i>Web Application Frontend & API Gateway</i>	5001	Memerlukan koneksi <i>healthy</i> ke DB sebelum memulai <i>Gunicorn</i> .
Background Scheduler	<i>Sentinel Engine</i> (Proses latar belakang)	N/A	Berjalan secara terpisah namun menggunakan <i>image</i> yang sama dengan <i>Web Server</i> .

Implementasi arsitektur berbasis kontainer ini menempatkan seluruh komponen utama dalam satu ekosistem yang terpadu dan terisolasi. Seluruh spesifikasi teknis, parameter lingkungan (*environment variables*), serta aturan orkestrasi antar-layanan didefinisikan menggunakan pendekatan *Infrastructure as Code (IaC)* melalui berkas *docker-compose.yml*. Untuk memastikan stabilitas



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

startup dan persistensi data, konfigurasi ini dipecah dan diuraikan berdasarkan fungsi masing-masing kontainer sebagai berikut:

1. Kontainer Basis Data Relasional (PostgreSQL)

Lapisan ini bertindak sebagai *Single Source of Truth* (SSOT) untuk menyimpan data kredensial terenkripsi dan konfigurasi sistem dengan konfigurasi seperti potongan kode pada Gambar 4.12 untuk konfigurasi bagian PostgreSQL `docker-compose.yaml`.

```

1 # Konfigurasi Bagian PostgreSQL docker-compose.yaml
2 postgres:
3   image: postgres:15
4   container_name: hub_postgres
5   environment:
6     POSTGRES_DB: orm_db
7     POSTGRES_USER: postgres
8     POSTGRES_PASSWORD: [REDACTED]
9   ports:
10    - "5432:5432"
11   healthcheck:
12     test: ["CMD-SHELL", "pg_isready -U $$POSTGRES_USER -d
13     $$POSTGRES_DB -h localhost"]
14     interval: 10s
15     timeout: 5s
16     retries: 5
17     start_period: 10s
18   volumes:
19     - postgres_data:/var/lib/postgresql/data
20   restart: unless-stopped
  
```

Gambar 4.13 Potongan Konfigurasi Bagian PostgreSQL `docker-compose.yaml`

Pengaturan paling krusial pada kontainer ini adalah blok *healthcheck*. Fungsi ini secara proaktif memverifikasi kesiapan basis data untuk menerima koneksi sebelum aplikasi utama (yang adalah Flask web-server) dijalankan. Hal ini mencegah terjadinya *error* akibat aplikasi mencoba mengakses *database* yang proses *booting*-nya belum selesai. Atribut *volumes* digunakan untuk memetakan penyimpanan internal kontainer ke penyimpanan *host*, menjamin persistensi data kredensial meskipun kontainer dimatikan atau di-*restart*.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

2. Kontainer Basis Data *In-Memory* (Redis)

Sebagai *Message Broker* dan lapisan *caching*, Redis membutuhkan alokasi port dan volume yang terdedikasi untuk menjaga performa asinkron sistem dengan konfigurasi seperti pada Gambar 4.14 berisi potongan kode konfigurasi bagian Redis docker-compose.yaml berikut.

```
1 # Konfigurasi Bagian Redis docker-compose.yaml
2 redis:
3   image: redis:7-alpine
4   container_name: hub_redis
5   ports:
6     - "6379:6379"
7   volumes:
8     - redis_data:/data
9   restart: unless-stopped
```

Gambar 4.14 Potongan Konfigurasi Bagian Redis docker-compose.yaml

Penggunaan *image* berbasis *alpine* dipilih untuk meminimalisir ukuran kontainer agar penggunaan memori efisien. Sama halnya dengan PostgreSQL, volume *redis_data* didefinisikan agar *state machine* status *uptime/downtime* komponen Wazuh tidak hilang ketika terjadi *restart* pada *server* fisik.

3. Kontainer Aplikasi Utama (Web Server & API Gateway)

Kontainer ini melayani interaksi pengguna melalui antarmuka web dan memproses permintaan API secara *on-demand* dengan konfigurasi seperti pada bagian kode di bawah.

```
1 # Konfigurasi Bagian Hub docker-compose.yaml
2 hub-app:
3   build: .
4   container_name: hub_flask_app
5   restart: unless-stopped
6   env_file: .env
7   ports:
8     - "5001:5001"
9   depends_on:
10    postgres:
```



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

11     condition: service_healthy
12     redis:
13     condition: service_started
14     command: >
15     sh -c "
16     flask db upgrade && \
17     python3 scripts/make_dev_cert.py && \
18     gunicorn --bind 0.0.0.0:5001 --certfile
instance/ssl/devcert.crt --keyfile instance/ssl/devcert.key
run:app"

```

Gambar 4.15 Potongan Konfigurasi Bagian Hub docker-compose.yaml

Konfigurasi `depends_on` dengan kondisi `service_healthy` memastikan aplikasi baru dijalankan setelah PostgreSQL siap menerima koneksi. Selanjutnya, blok `command` secara otomatis menjalankan migrasi basis data, membuat sertifikat SSL pengembangan, dan memulai Gunicorn sebagai WSGI HTTP Server secara berurutan.

4. Kontainer Background Scheduler (*Sentinel Engine*)

Sebagai upaya mengurangi risiko peningkatan beban thread pada proses web utama, komponen scheduler diimplementasikan dalam kontainer tersendiri sehingga proses latar belakang dapat berjalan secara independen dari layanan web dengan konfigurasi seperti bagian kode pada Gambar 4.15

```

1  # Konfigurasi Bagian Background Scheduler docker-compose.yaml
2  hub-scheduler:
3    build: .
4    container_name: hub_scheduler
5    restart: unless-stopped
6    env_file: .env
7    depends_on:
8      postgres:
9        condition: service_healthy
10     redis:
11     condition: service_started
12     command: python scheduler.py

```

Gambar 4.16 Potongan Konfigurasi Background Scheduler docker-compose.yaml



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Kontainer ini menggunakan *image* yang identik dengan kontainer Hub di poin sebelumnya. Tujuannya adalah memberikan efisiensi komputasi karena Docker *engine* hanya perlu melakukan proses *build* satu kali. Perbedaan esensial terletak pada instruksi *command* akhir, di mana kontainer ini difokuskan secara eksklusif untuk menjalankan skrip *scheduler.py* yang bertugas melakukan *polling* status Wazuh dan mengirimkan *payload* notifikasi Telegram secara independen tanpa memblokir lalu lintas jaringan aplikasi web.

5. Definisi Volume *Global*

Sebagai mekanisme penyimpanan data yang bersifat permanen, ruang penyimpanan lokal didefinisikan pada bagian akhir berkas konfigurasi.

```
1 volumes
2 postgres_data: {path to dir}
3 redis_data: {path to dir}
```

Gambar 4.17 Konfigurasi Bagian Volume *Global* docker-compose.yml

Deklarasi volume terkelola ini memastikan Docker *engine* mengamankan data konfigurasi hub manajemen dan *state* operasional, sehingga mencegah terjadinya kebocoran memori atau hilangnya informasi saat kontainer dihancurkan dan dibuat ulang.

4.3.2 Implementasi Hub Manajemen Konfigurasi Multi-Wazuh

Sistem Hub Manajemen Konfigurasi Multi-Wazuh bertindak sebagai *Single Source of Truth* (SSOT) yang memusatkan seluruh konfigurasi infrastruktur keamanan Wazuh yang ada. Fungsionalitas ini memungkinkan Administrator untuk mendaftarkan, mengubah, dan menghapus koneksi ke *node* Wazuh (*Manager* dan *Indexer*) secara terpusat. Untuk memastikan integritas arsitektur, implementasi modul ini dibangun dengan mengintegrasikan konektor API, lapisan keamanan kriptografi, dan logika validasi yang ketat. Detail pengimplementasian tersebut dijabarkan berdasarkan kategori implementasi.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

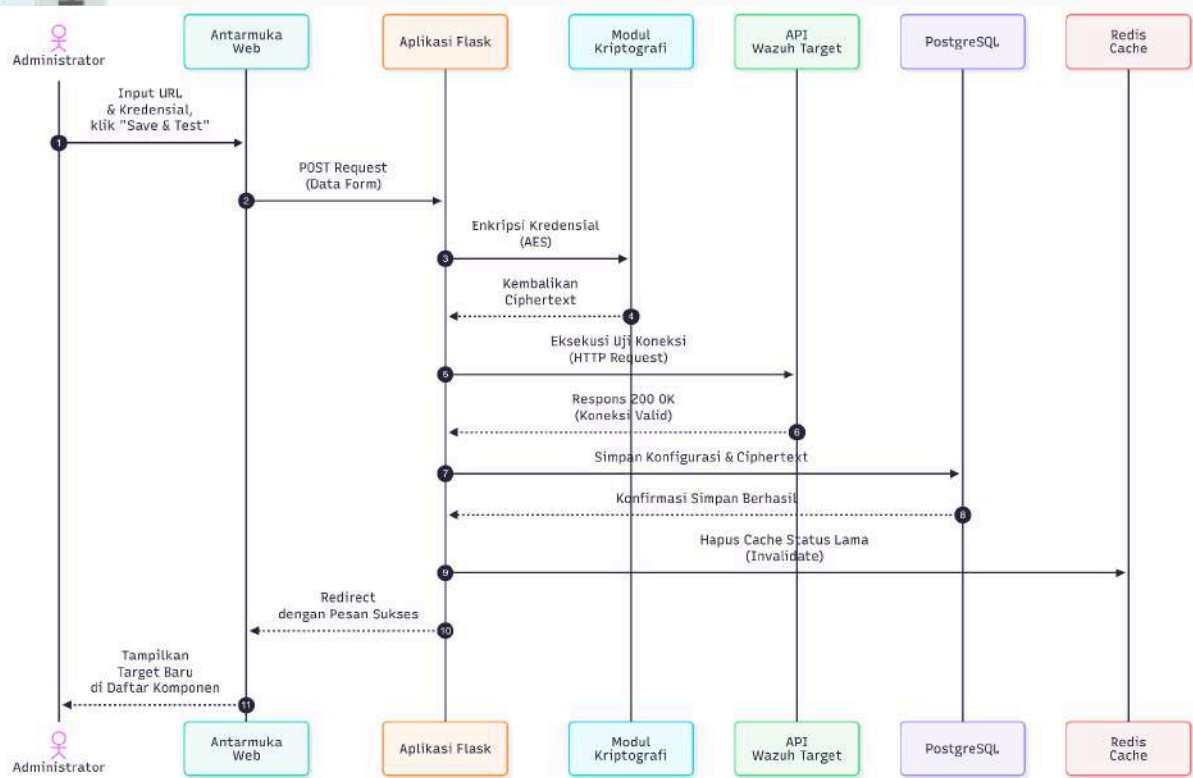
A. Alur Sekuensial Operasional

Data yang disimpan dalam hub manajemen konfigurasi harus berupa data yang valid dan aman. Untuk mencegah penyimpanan data konfigurasi yang tidak valid (seperti alamat URL mati atau kata sandi yang salah), sistem mengimplementasikan pola *Validate Before Save*. Alur kerja fungsionalitas ini dimodelkan ke dalam dua skenario diagram sekuensial:

1. Penambahan Komponen (Skenario Berhasil)

Skenario ini terjadi ketika Administrator memasukkan kredensial yang valid dan *server* target dapat dijangkau oleh jaringan. Penjelasan ringkas terkait gambaran besar alur ini adalah sebagai berikut:

- i. Inisiasi & Enkripsi: Saat data dikirimkan melalui *form* (Langkah 1-2), *backend* Flask tidak langsung melakukan proses validasi. Sebagai langkah awal, sistem secara otomatis meneruskan *password* ke modul kriptografi untuk dienkripsi di dalam memori (Langkah 3-4), sehingga data sensitif tidak pernah diproses dalam bentuk *plaintext*.
- ii. Validasi Eksternal: Setelah proses enkripsi selesai, *backend* mengirimkan permintaan ke API Wazuh tujuan menggunakan kredensial yang telah disiapkan (Langkah 5). Respons 200 OK yang diterima (Langkah 6) menjadi indikasi bahwa proses autentikasi berhasil, sehingga sistem dapat mengonfirmasi bahwa kredensial yang dimasukkan valid dan dapat digunakan.
- iii. Persistensi & Optimasi: Selanjutnya, data konfigurasi beserta kredensial yang telah dienkripsi disimpan ke PostgreSQL (Langkah 7) agar dapat digunakan kembali pada proses berikutnya. Untuk memastikan dashboard segera menampilkan informasi terbaru, sistem secara proaktif menghapus (*invalidate*) data status lama yang tersimpan di Redis (Langkah 9) sebelum mengembalikan halaman kepada Administrator, sehingga perubahan konfigurasi dapat langsung tercermin pada antarmuka.



Gambar 4.18 Diagram Alur Sekuensial Penambahan Komponen Berhasil

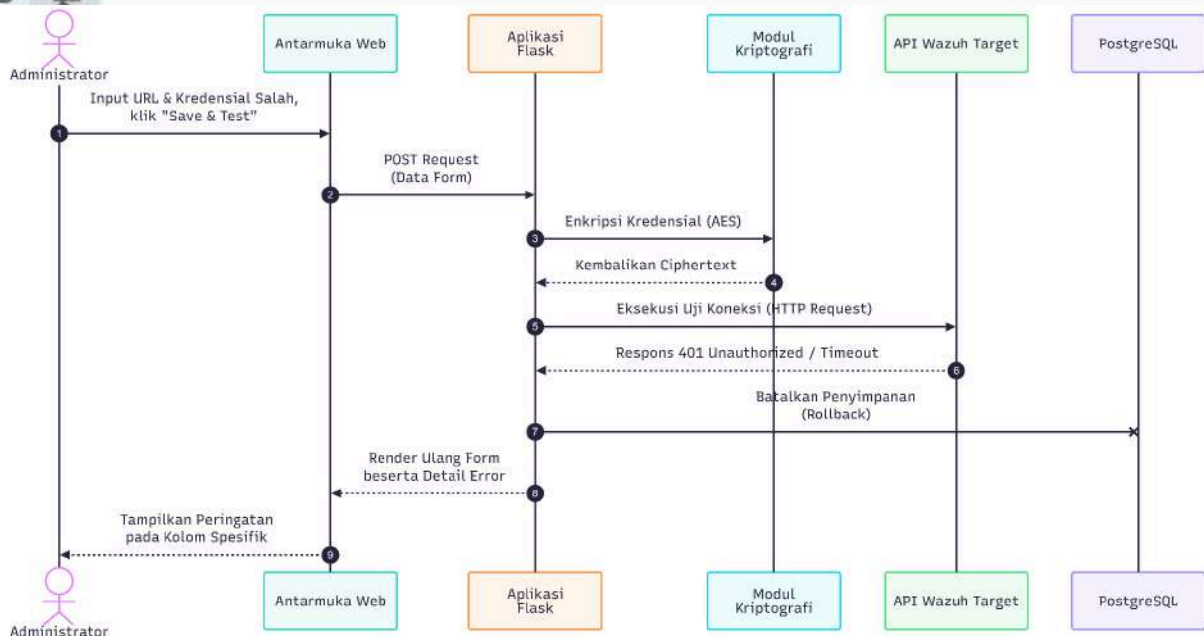
2. Penambahan Komponen (Skenario Gagal)

Skenario ini menjamin integritas sistem dengan menolak penyimpanan data jika terjadi kesalahan jaringan atau autentikasi. Penjelasan ringkas terkait gambaran besar alur ini adalah sebagai berikut:

- Siklus awal berjalan identik dengan skenario berhasil (1) hingga tahap pengujian koneksi (Langkah 1-5). Namun, ketika API Wazuh target mengembalikan respons penolakan seperti 401 *Unauthorized* atau koneksi mengalami *Timeout* (Langkah 6), alur sistem berubah secara fundamental:
- Backend* langsung membatalkan proses dan memutus jalur ke basis data PostgreSQL (Langkah 7). Data kredensial yang salah akan langsung dibuang dari memori.
- Umpan Balik Kontekstual: Sistem mengevaluasi jenis kode *error* yang diterima dan meneruskannya ke antarmuka web (Langkah 8). Jika *error*



adalah 401, sistem akan menyorot kolom *Username/Password* di layar Administrator. Jika *error* adalah *Timeout*, sistem akan menyorot kolom URL. Hal ini memberikan panduan perbaikan yang presisi bagi Administrator.



Gambar 4.19 Diagram Alur Sekuensial Penambahan Komponen Gagal

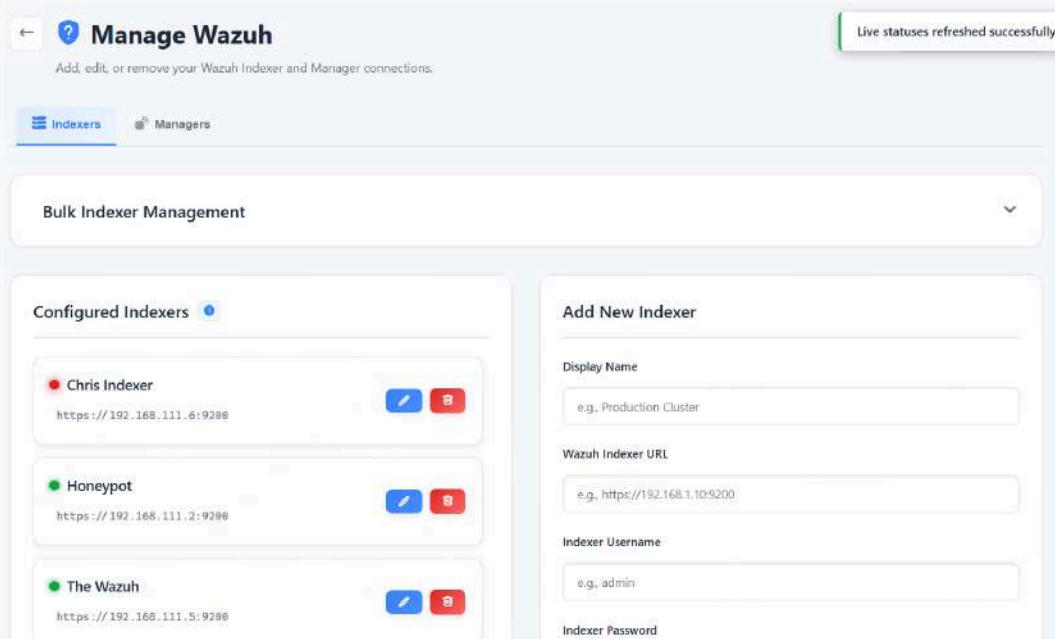
Seluruh proses penambahan, validasi, dan agregasi data komponen dilakukan melalui satu antarmuka terpadu (*unified interface*) sehingga administrator tidak perlu berpindah ke beberapa halaman atau aplikasi. Seperti ditunjukkan pada Gambar 4.18, antarmuka Manajemen Komponen Infrastruktur (Wazuh Indexer) dibagi menjadi dua area utama.

Sisi kanan menyediakan formulir untuk memasukkan kredensial dan informasi koneksi komponen yang akan didaftarkan, sedangkan sisi kiri menampilkan daftar komponen yang telah berhasil divalidasi dan disimpan di dalam basis data. Tata letak ini memungkinkan administrator melakukan konfigurasi sekaligus memantau hasil validasi dalam satu tampilan yang terintegrasi.



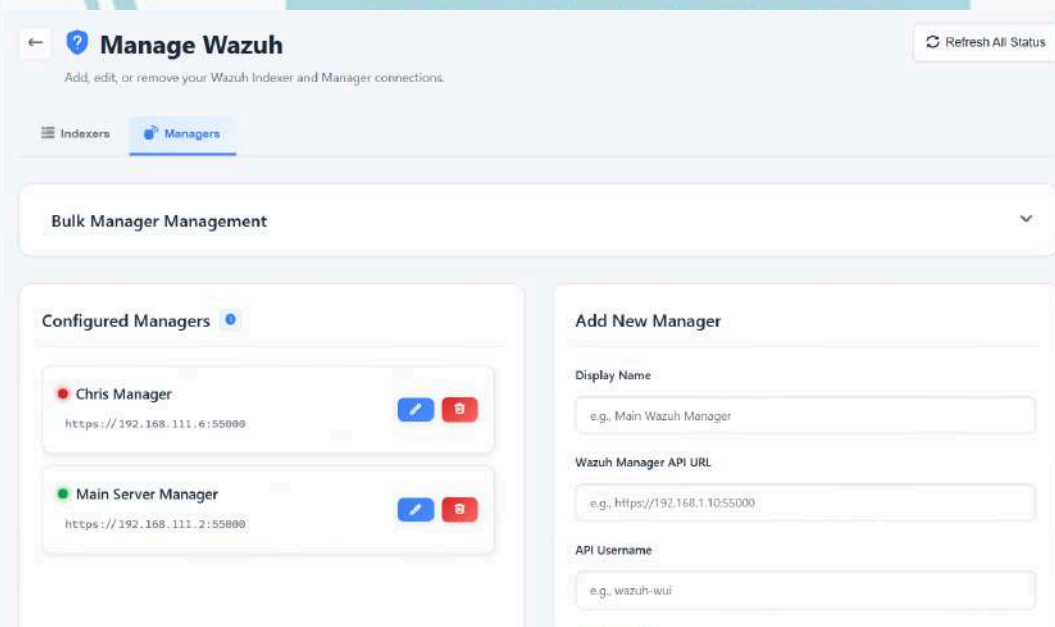
Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 4.20 Antarmuka Manajemen Komponen Infrastruktur (Wazuh Indexer)

Tampilan yang sama juga digunakan pada antarmuka manajemen komponen lainnya yakni Wazuh Manager dengan perbedaan berupa *field* autentikasi yang memasukkan API *username* & API *password*.



Gambar 4.21 Antarmuka Manajemen Komponen Infrastruktur (Wazuh Manager)

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Sebelum sistem konektor diimplementasikan, interaksi dengan komponen infrastruktur Wazuh mengharuskan analis SOC untuk melakukan *query* secara manual melalui antarmuka baris perintah (CLI). Proses ini memperlihatkan kompleksitas teknis yang signifikan, mengingat Wazuh *Manager* dan Wazuh *Indexer* menerapkan mekanisme autentikasi dan pengembalian struktur data yang secara fundamental berbeda.

Sebagai contoh, komunikasi dengan *Wazuh Manager* pada port 55000 mensyaratkan mekanisme *two-step authentication*, dengan terlebih dahulu mengirimkan request ke *endpoint* `/security/user/authenticate`. Administrator tidak bisa langsung menarik data, melainkan harus mengirimkan kredensial dengan command `curl -u <USERNAME>:<PASSWORD> -k -X POST "https://<IP_MANAGER>:55000/security/user/authenticate"` untuk memicu *endpoint* autentikasi agar sistem mengembalikan JSON Web Token (JWT) yang panjang, sebagaimana ditunjukkan pada Gambar 4.22 untuk request dengan kredensial salah.

```
chris@kali:~$ curl -X GET "https://192.168.111.6:9200/wazuh-alerts*/_search" -u admin:wrong -k -H 'Content-Type: application/json' -d '{
  "size": 1,
  "sort": [
    {
      "@timestamp": {
        "order": "desc"
      }
    }
  ]
}'
chris@kali:~$ tel-rii wazuh-wui:wrongpass -k -X POST "https://192.168.111.2:55000/security/user/authenticate" {"title": "Unauthorized", "detail": "Invalid credentials"}chris@kali:~$
```

Gambar 4.23 menunjukkan hasil request dengan kredensial benar. *Token* ini kemudian harus disalin dan disematkan secara manual ke dalam *header authorization* pada request selanjutnya.

```
chris@kali:~$ curl -u [REDACTED] -k -X POST "https://192.168.111.2:55000/security/user/authenticate"
{"data": {"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3YXplcmFZICI6IldhenVoEjEFQSSSRVNUiwiwibmJmIjozczNDQ3NTYxLGlleHAiojEjNzc0NDg0NjEsInN1YiI6IndhenVoXdaSI6InJlb19hcyl6ZmFsc2UiInJlYWxfcm9sZXMiOlsxXSwicmJyY19tb2RlIjoid2hpdGUufQ.AznHmp8ynORYYZDU7wlAx4Wg6BP_V_LgjUz08uWC_HqaEsLn0aJQBpM5o75og0MdSw_CjkmLS3aX6vL152IdAntVa2q-qmV5eQxynt_PPtI3Q8S5T53e9eq24t0CqgPeEnHbn14CDCSes_K_BX7-luSViD8YIVZebDuUpJu4-hndJ"}; "error": 0}chris
```

Gambar 4.23 *Raw* Response Generasi *Token* JWT pada *Wazuh Manager* via CLI



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Di sisi lain, penarikan data log atau *alert* dari Wazuh *Indexer* pada port 9200 menggunakan Basic *Authentication*. Meskipun langkah autentikasinya lebih singkat, *raw* response yang dikembalikan oleh *server* berupa struktur objek JSON yang sangat padat, bertingkat (*nested*), dan sulit dibaca secara sekilas (*cluttered*), seperti yang diilustrasikan pada Gambar 4.24.

```
chris@~$ curl -X GET "https://192.168.111.6:9200/wazuh-alerts-*/_search" -u ad: -k -H 'Content-Type: application/json' -d '{
  "size": 1,
  "sort": [
    {
      "@timestamp": {
        "order": "desc"
      }
    }
  ]
}'
{"took":57,"timed_out":false,"_shards":{"total":615,"successful":615,"skipped":0,"failed":0},
"hits":{"total":{"value":10000,"relation":"gte"},"max_score":null,"hits":[{"_index":"wazuh-alerts-4.x-2026.04.29","_id":"jSoW2J0BnDLnThKrAE0","_score":null,"_source":{"predecoder":{"hostname":"lst-thread-intel-risk","program_name":"sshd","timestamp":"Apr 29 07:14:05"},"agent":{"ip":"127.0.0.1","name":"ubuntu-chris","id":"001"},"manager":{"name":"wazuh.manager"},"data":{"srcuser":"chris","uid":"0","dstuser":"chris(uid=1002)","rule":{"mail":false,"level":3,"pci_dss":["10.2.5"],"hipaa":["164.312.b"],"tsc":["CC6.8","CC7.2","CC7.3"],"description":"PAM: Login session opened.","groups":["pam","syslog"],"authentication_success":["nist_800_53":["AU.14","AC.7"],"gdpr":["IV_32.2"],"firedtimes":1,"mitre":{"technique":["Valid Accounts"],"id":["T1078"],"tactic":["Defense Evasion","Persistence","Privilege Escalation","Initial Access"],"id":["5501"],"gpg13":["7.8","7.9"]},"decoder":{"parent":"pam","name":"pam"},"full_log":["Apr 29 07:14:05 lst-thread-intel-risk sshd[1464320]: pam_unix(sshd:session): session opened for user chris(uid=1002) by chris(uid=0)","input":{"type":"log"},"@timestamp":"2026-04-29T07:14:07.920Z","location":"journald","id":"1777446847.457","timestamp":"2026-04-29T07:14:07.920+0000"},"sort":["1777446847920"]}]}}chris@lst-thread-intel-risk:~$
```

Gambar 4.24 Raw Response Data Log dari Wazuh *Indexer* via CLI

Berdasarkan kedua metode manual di atas, terlihat jelas bahwa penarikan dan pembacaan output mentah di terminal sangat tidak efisien dan rentan terhadap *human error*. Memaksa analis untuk terus-menerus memilah baris JSON di terminal secara manual akan memicu beban kognitif dan context-switching yang masif. Hal ini berbanding terbalik dengan tujuan optimalisasi *breakout time* yang membutuhkan kecepatan analisis.

Oleh karena itu, seluruh kompleksitas pertukaran data dan autentikasi ini diabstraksi ke dalam lapisan konektor (*connector layer*) pada aplikasi Hub Manajemen. Sistem bertugas mengambil alih proses request, menangani kredensial *in-memory*, dan melakukan *parsing* JSON di latar belakang (*backend*). Implementasi konektor untuk Wazuh *Indexer* yang berbasis HTTP Basic *Authentication* terdefinisi pada potongan kode di Gambar 4.25.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

1  # Gambar 4.25 Potongan Kode Konektor Wazuh Indexer (HTTP Basic
Authentication)
2  def test_wazuh_connection(indexer, encryption_key, logger):
3      # 1. Dekripsi kredensial secara in-memory menggunakan masterkey
4      password = indexer.get_decrypted_password(encryption_key)
5      if not password:
6          return (False, "auth_error", "Gagal mendekripsi kredensial.")
7
8      try:
9          # 2. Membuka sesi HTTP persisten untuk efisiensi eksekusi
10         with requests.Session() as session:
11             session.auth = (indexer.username, password)
12
13             # 3. Melakukan HTTP GET request dengan batas waktu (timeout) 3
detik
14             response = session.get(indexer.url, verify=False, timeout=3)
15
16             # 4. Memvalidasi status HTTP dan struktur JSON respons secara
spesifik
17             if response.status_code == 200:
18                 data = response.json()
19                 if "cluster_name" in data:
20                     return (True, "success", f"Berhasil terhubung ke
{indexer.name}.")
21                 return (False, "connection_error", "Respons bukan
merupakan API Wazuh.")
22
23             elif response.status_code in [401, 403]:
24                 return (False, "auth_error", "Autentikasi ditolak oleh
server.")
25
26             # 5. Menangani eksepsi jaringan agar proses komputasi tidak terhenti
(crash)
27             except requests.exceptions.RequestException as e:
28                 return (False, "connection_error", f"Koneksi jaringan terputus:
{e}")

```

Gambar 4.25 Potongan Kode Konektor Wazuh Indexer (HTTP Basic Authentication)

Berbeda dengan *Indexer*, *Wazuh Manager* mensyaratkan mekanisme *two-step authentication* menggunakan JSON Web Token (JWT). Meminta JWT baru setiap kali sistem melakukan pemantauan akan membebani komputasi *server*. Oleh karena itu, implementasi ini memanfaatkan Redis untuk melakukan *caching* terhadap *token*.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

1  # Gambar 4.26 Potongan Konektor Wazuh Manager (JWT & Caching)
2  def _get_auth_token(manager, app, force_refresh=False):
3      redis_client = app.redis
4      token_cache_key = f"sys:manager:{manager.id}:token"
5
6      # 1. Memeriksa ketersediaan JWT yang masih valid di Redis Cache (Cache
HIT)
7      if not force_refresh and redis_client:
8          cached_token = redis_client.get(token_cache_key)
9          if cached_token:
10             return cached_token.decode("utf-8")
11
12     # 2. Mendekripsi kredensial jika token tidak ditemukan di memori
(Cache MISS)
13     creds =
manager.get_decrypted_credentials(app.config["ENCRYPTION_KEY"])
14     if not creds: return None
15     username, password = creds
16
17     # 3. Meminta JWT baru ke endpoint autentikasi manajer
18     auth_url = f"{manager.url}/security/user/authenticate"
19     try:
20         response = requests.post(auth_url, auth=(username, password),
verify=False, timeout=10)
21         response.raise_for_status()
22         token = response.json().get("data", {}).get("token")
23
24     # 4. Menyimpan token baru ke Redis dengan batas waktu (TTL) 14
menit
25     if token and redis_client:
26         redis_client.setex(token_cache_key, 840, token)
27         return token
28
29     except requests.exceptions.RequestException:
30         return None
31
32     def test_wazuh_manager_connection(app, manager, force_refresh=False):
33         # 5. Mengambil JWT yang valid dari fungsi helper di atas
34         token = _get_auth_token(manager, app, force_refresh=force_refresh)
35         if not token:
36             return (False, "auth_error", "Gagal mendapatkan token
autentikasi.")
37
38         # 6. Menyisipkan JWT ke dalam Header HTTP Authorization (Bearer)
39         headers = {"Authorization": f"Bearer {token}"}
40         status_url = f"{manager.url}/manager/status"
41
42         try:
43             response = requests.get(status_url, headers=headers, verify=False,
timeout=10)
44             response.raise_for_status()

```



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

45
46     if response.json().get("error") == 0:
47         return (True, "success", f"Berhasil membaca status manajer.")
48
49     except requests.exceptions.HTTPError as e:
50         # 7. Penanganan reaktif: Paksa pembaruan token jika mendapat
51         respons 401 Unauthorized
52         if e.response.status_code == 401 and not force_refresh:
53             return test_wazuh_manager_connection(app, manager,
54             force_refresh=True)
55         return (False, "connection_error", "Gagal mengautentikasi Token.")
56

```

Gambar 4.26 Potongan Konektor Wazuh Manager (JWT & Caching)

C. Alur Penyimpanan dan Keamanan Kredensial

Sebelum konektor API di bagian atas dapat bekerja, sistem harus memastikan bahwa data kredensial yang diinputkan oleh pengguna dikelola dengan aman. Sebagai implementasi dari prinsip Security by Design, kata sandi dilarang keras disimpan dalam format teks terang (*plaintext*).

Sistem mengatasi hal ini dengan menerapkan enkripsi simetris menggunakan algoritma AES (*Advanced Encryption Standard*). Proses enkripsi dienkapsulasi secara otomatis di dalam *layer* model basis data (*SQLAlchemy*).

```

1  # Potongan Kode Model Data WazuhManager
2  from cryptography.fernet import Fernet
3  from sqlalchemy.ext.hybrid import hybrid_property
4
5  class WazuhManager(db.Model):
6      # 1. Field fisik di database yang secara spesifik menyimpan data
7      # biner terenkripsi
8      _encrypted_password = db.Column(db.LargeBinary, nullable=False)
9
10     @staticmethod
11     def _get_crypter():
12         # 2. Mengambil Kunci Master (ENCRYPTION_KEY) dari konfigurasi
13         # server yang aman
14         key = current_app.config.get("ENCRYPTION_KEY")
15         return Fernet(key.encode())
16
17     @hybrid_property
18     def password(self):
19         # 3. Getter: Mendekripsi data biner menjadi teks kembali
20         # hanya saat dibutuhkan di memori

```



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

18         if not self._encrypted_password: return None
19         return
20     self._get_crypter().decrypt(self._encrypted_password).decode()
21
22     @password.setter
23     def password(self, plaintext_password):
24         # 4. Setter: Mengenkripsi input teks menjadi format
25         ciphertext (LargeBinary)
26         self._encrypted_password = \
27         self._get_crypter().encrypt(plaintext_password.encode())

```

Gambar 4.27 Potongan Kode Model Data *WazuhManager*

Setiap kali aplikasi menerima *password* baru, objek kriptografi akan langsung mengubah *string* tersebut menjadi *ciphertext* menggunakan *setter*. Alhasil, jika basis data berhasil diretas, kredensial infrastruktur akan tetap aman dan tidak dapat dibaca. Gambar 4.28 dan Gambar 4.29 menunjukkan kolom *encrypted_password* pada *database Wazuh Indexer* dan tambahan kolom *encrypted_username Manager* yang sudah dalam bentuk *ciphertext*.

id	name	url	username	encrypted_password	is_active
2	Ubuntu 2	https://192.168.111.7:9200	admin	gAAAAABo9dT1dAX10ZbtajcgDTea-FNTa0SmA3	true
5	Tama Wazuh	https://192.168.111.5:9200	admin	gAAAAABpIWY_7z5gcZNI6UvABow4yXPDVVOjQi	true
6	Chris Indexer	https://192.168.111.6:9200	admin	gAAAAABpoQ0fk-tu4RK0cNaCl69zk1UK5Ce0cnvK	true
1	Honeypot	https://192.168.111.2:9200	admin	gAAAAABo9bRpaBRASKL40cPZBhcW79zeReavenI	true

Gambar 4.28 Tampilan *Table wazuh_indexers*

id	name	url	encrypted_username	encrypted_password	is_active
1	Main Server Manager	https://192.168.111.2:55000	gAAAAABo9bTuvrs8Jvv90K	gAAAAABo9bTus6AWyiRVyZ_IYUzt	true
3	Chris Manager	https://192.168.111.6:55000	gAAAAABp8F4ahM1Kjz20C	gAAAAABp8F4aBYfRIhOZJcfONmX	true

Gambar 4.29 Tampilan *Table wazuh_managers*

D. Alur Validasi dan Penanganan Kesalahan (*Controller*)

Seluruh mekanisme konektor (B) dan keamanan kredensial (C) tersebut diorkestrasi di dalam lapisan *Controller*. Lapisan ini memiliki beberapa tugas: menerima input dari antarmuka, menginisiasi pengujian, dan pada akhirnya memutuskan apakah konfigurasi diizinkan masuk ke basis data (PostgreSQL).



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

1  # Potongan Kode Controller Hub Multi-Wazuh
2  if manager_form.validate_on_submit():
3      # 1. Menerima input kredensial dari form (Enkripsi otomatis
      tereksekusi di sini)
4      new_manager = WazuhManager(
5          name=manager_form.name.data,
6          url=manager_form.url.data,
7          username=manager_form.username.data,
8          password=manager_form.password.data,
9      )
10
11     # 2. Memanggil fungsi konektor API eksternal untuk pengujian
    validitas
12     app = current_app._get_current_object()
13     is_ok, status_code, message = test_manager_connection(app,
    new_manager)
14
15     if is_ok:
16         try:
17             # 3. Mengeksekusi penyimpanan ke basis data persisten
18             db.session.add(new_manager)
19             db.session.commit()
20
21             # 4. Menghapus cache status lama di Redis agar UI segera
    terbaru
22             if current_app.redis:
23                 current_app.redis.delete(MANAGER_STATUS_CACHE_KEY)
24
25             flash(f"Manager '{new_manager.name}' berhasil
    ditambahkan.", "success")
26             return redirect(url_for("wazuh.manage",
    active_tab="managers"))
27
28         except IntegrityError:
29             # Menangani duplikasi nama komponen di dalam database
30             db.session.rollback()
31             flash("Nama atau URL manager sudah digunakan.", "danger")
32
33     else:
34         # 5. Isolasi kegagalan: Memetakan tipe error jaringan ke
    kolom form yang spesifik
35         error_message = f"Uji Koneksi Gagal: {message}"
36         if status_code == "auth_error":
37             manager_form.username.errors.append(error_message)

```

Gambar 4.30 Potongan Kode *Controller* Hub Multi-Wazuh

Implementasi ini mengadopsi arsitektur *fail-fast*, di mana sistem akan langsung membatalkan penyimpanan jika uji koneksi gagal. Pada Langkah 5 di



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

gambar, sistem merespons kegagalan jaringan dengan memetakan kode *error* secara spesifik. Jika status yang dikembalikan adalah *auth_error*, pesan kesalahan akan disematkan langsung di bawah kolom *Username*. Hal ini memberikan panduan perbaikan yang sangat intuitif bagi Administrator.

E. Manajemen *Read, Update, Delete* Komponen dan Mekanisme *Refresh*

Setelah komponen berhasil didaftarkan, sistem menyediakan fungsionalitas untuk membaca (*Read*), memperbarui (*Update*), dan menghapus (*Delete*) konfigurasi. Tantangan utama pada fase ini adalah menjaga sinkronisasi antara data yang ada di basis data PostgreSQL dengan status operasional berbasis cache Redis yang ditampilkan kepada pengguna, tanpa membebani jaringan dengan request yang berlebihan.

1. Logika Pembaruan (*Edit*) dan Penghapusan (*Delete*)

Pada operasi pembaruan, sistem mengizinkan Administrator untuk mengubah detail konfigurasi. Prinsip *Validate Before Save* kembali diterapkan di sini; sistem akan menguji ulang koneksi baru sebelum menimpa data lama. Khusus untuk operasi edit, pembaruan kata sandi bersifat opsional.

Hal yang paling krusial dari operasi pembaruan dan penghapusan adalah proses invalidasi *cache*. Setiap kali terjadi perubahan arsitektur, cache status yang ada di memori harus segera dihancurkan agar tidak terjadi status data usang di layar pengguna. Potongan logika untuk melakukan edit dan hapus tertera seperti pada gambar 4.31.

```

1  # Potongan Kode Logika Edit dan Delete Hub Multi-Wazuh
2  if edit_form.validate_on_submit():
3      # 1. Memperbarui atribut komponen (Password bersifat opsional)
4      editing_item.name = edit_form.name.data
5      editing_item.url = edit_form.url.data
6      if edit_form.password.data:
7          editing_item.password = edit_form.password.data
8
9      # 2. Re-validasi: Menguji ulang koneksi dengan konfigurasi baru
10     is_ok, status_code, message = test_connection(editing_item)
11     if is_ok:
12         db.session.commit()
13     # 3. Invalidasi Cache: Hapus status lama di Redis

```



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

14         if current_app.redis:
15             current_app.redis.delete(STATUS_CACHE_KEY)
16         return redirect(url_for("wazuh.manage"))
17
18 @bp.route("/delete/<int:item_id>", methods=["POST"])
19 def delete_item(item_id):
20     # 4. Operasi Delete: Hapus dari database dan hancurkan cache
21     item = Component.query.get_or_404(item_id)
22     db.session.delete(item)
23     db.session.commit()
24     if current_app.redis:
25         current_app.redis.delete(STATUS_CACHE_KEY)

```

Gambar 4.31 Potongan Kode Logika Edit dan *Delete* Hub Multi-Wazuh

Pada tampilannya di antarmuka seperti yang terlihat pada gambar 4.32 setiap kali Administrator menginisiasi operasi *Update*, sistem memberikan umpan balik kontekstual berupa indikator *toast* yang menandakan komponen spesifik yang sedang dimodifikasi.

Gambar 4.32 Tampilan Notifikasi *Toast* saat Mengedit Komponen

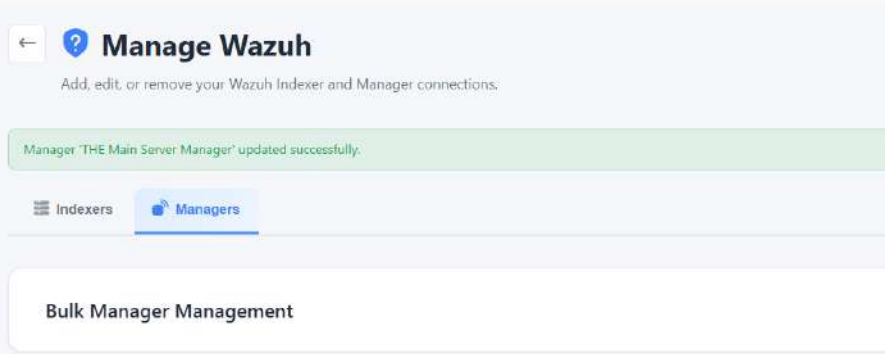
Apabila proses validasi koneksi berhasil, sistem akan secara otomatis menyimpan perubahan konfigurasi dan menampilkan pesan konfirmasi sebagai indikator bahwa pembaruan telah berhasil diterapkan. Konfirmasi ini memberikan umpan balik langsung kepada administrator bahwa kredensial dan koneksi yang



Hak Cipta :

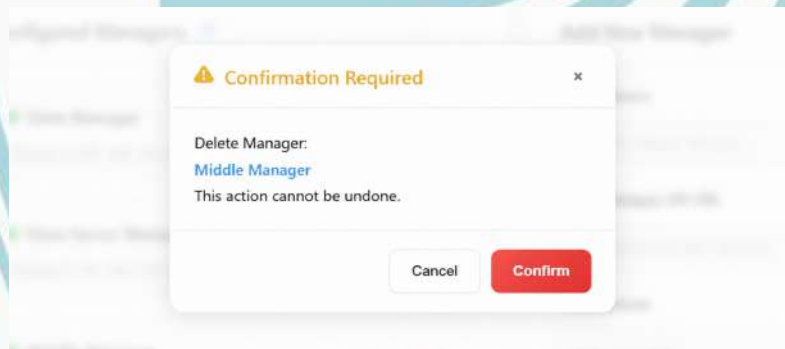
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

digunakan telah terverifikasi serta siap digunakan oleh sistem. Tampilan pesan keberhasilan tersebut ditunjukkan pada Gambar 4.33.



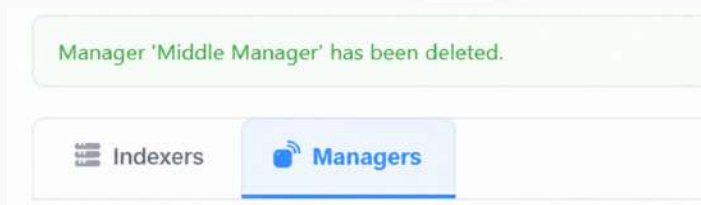
Gambar 4.33 Tampilan Notifikasi *Toast* saat *Update* Komponen Berhasil

Sementara itu, untuk operasi penghapusan (*Delete*), sistem menerapkan prinsip *safe-delete*. Dibandingkan langsung menghapus data saat tombol ditekan, antarmuka akan menampilkan *modal* konfirmasi untuk mencegah risiko penghapusan secara tidak sengaja seperti pada gambar 4.34.



Gambar 4.34 Tampilan *Modal* untuk Konfirmasi Penghapusan Komponen

Saat penghapusan berhasil maka akan muncul *toast* berwarna hijau mengkonfirmasi komponen apa yang dihapus beserta namanya.



Gambar 4.35 Tampilan *Toast* Notifikasi Saat Berhasil Menghapus Komponen



2. Mekanisme *Caching* dan Tarik Status (*Read*)

Saat memuat daftar komponen, sistem harus menampilkan status konektivitas seketika (*online/offline*). Alih-alih melakukan *ping* ke *server* Wazuh setiap kali halaman dimuat, *backend* menyimpan hasil *ping* terakhir di Redis dengan batas waktu (Time-To-Live / TTL) selama 10 detik.

Logika *frontend* berbasis JavaScript dibangun untuk merespons status dari cache ini dan memberikan umpan balik yang transparan kepada Administrator melalui notifikasi *toast*.

```

1 // Potongan Kode Logika Frontend Refresh
2 async function _fetchAllStatuses(force = false) {
3   try {
4     // 1. Meminta status ke endpoint internal API
5     const response = await
fetch(`/api/managers/status?force=${force}`);
6     const result = await response.json();
7
8     // 2. Memperbarui indikator UI (Hijau/Merah) berdasarkan
hasil
9     updateStatusIndicators(result.data, 'manager');
10
11    // 3. Logika Umpan Balik Notifikasi berdasarkan Sumber Data
12    if (force || result.source === 'api') {
13      // Jika data diambil langsung dari API eksternal
14      showToast('Live statuses refreshed successfully.',
'success');
15    } else {
16      // Jika data diambil dari cache Redis (umur < 10 detik)
17      showToast('Status loaded from recent cache.', 'info',
2000);
18    }
19  } catch (error) {
20    showToast("An error occurred while fetching statuses.",
"error");
21  }
22 }

```

Gambar 4.36 Potongan Kode Logika *Frontend Refresh*

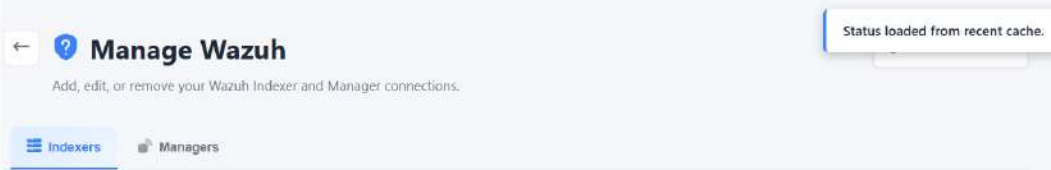
Berdasarkan logika kode Gambar 4.36, ketika Administrator menekan tombol *refresh*, sistem mengevaluasi informasi keterbaruan data. Jika permintaan terjadi di dalam jendela waktu 10 detik sejak *refresh* terakhir, sistem tidak akan membebani jaringan, melainkan mengambil data dari Redis dan memunculkan



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

notifikasi "Status loaded from recent cache" seperti terlihat pada Gambar 4.37.



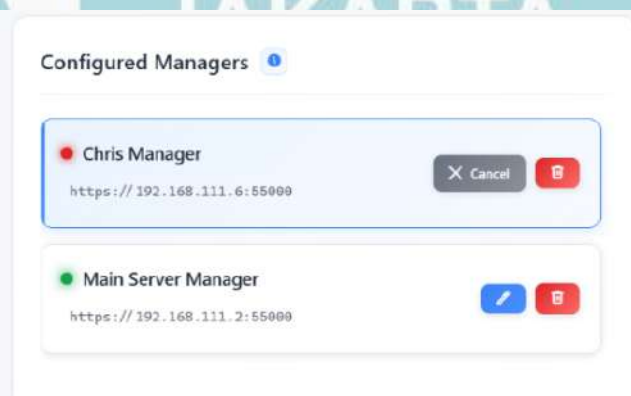
Gambar 4.37 Tampilan Notifikasi *Toast* saat *Refresh* dari Cache

Jika sudah melewati batas TTL, sistem akan melakukan pengujian koneksi aktual ke API Wazuh dan menampilkan notifikasi "Live statuses refreshed successfully" seperti pada Gambar 4.38.



Gambar 4.38 Tampilan Notifikasi *Toast* saat *Refresh* Live Status

Hasil dari mekanisme *health check* asinkron dan *caching* ini langsung direfleksikan pada daftar komponen di layar pengguna. Sistem menggunakan indikator visual yang intuitif: warna hijau menandakan *endpoint* berhasil dijangkau (*connected*), sedangkan warna merah mengindikasikan bahwa *server* mengalami *timeout* atau tidak dapat diakses (*unreachable*).



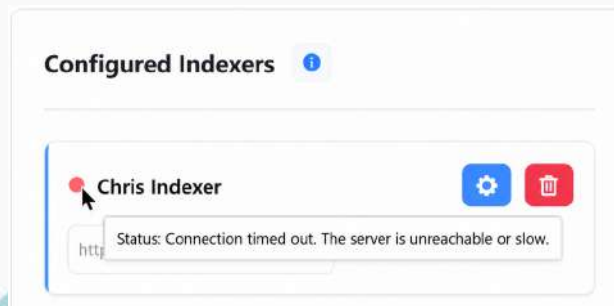
Gambar 4.39 Tampilan Status Indikator saat Komponen Tersambung



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Saat komponen berubah status akibat *connection timed out* ataupun penyebab lain, Indikator merah akan berisi toast notifikasi status seperti Gambar 4.40.



Gambar 4.40 Tampilan Status Indikator saat Komponen Terputus

4.3.3 Implementasi Hub Agregasi Multi-Agen Wazuh

Infrastruktur keamanan berskala besar sering kali memiliki ratusan hingga ribuan agen yang terdistribusi di berbagai kluster *Wazuh Manager*. Hal ini adalah bentuk nyata dari permasalahan fragmentasi operasional, di mana analis harus membuka banyak tab terpisah untuk memantau seluruh agen. Untuk mengatasi masalah ini, sistem mengimplementasikan modul Hub Agregasi Multi-Agen Wazuh yang bertugas menarik, menggabungkan, dan menyajikan data agen wazuh dari seluruh manajer ke dalam satu antarmuka terpusat (*Single Pane of Glass*).

A. Alur Penarikan Data Paralel *Agent Wazuh*

Tantangan utama dalam melakukan pengumpulan dari berbagai sumber eksternal adalah latensi antar satu sumber ke lainnya. Jika sistem mengonfigurasi lima *Wazuh Manager* dan masing-masing membutuhkan waktu dua detik untuk merespons, penarikan data secara *sequential* akan memakan waktu sepuluh detik, yang mana akan menurunkan performa tim.

Secara *default*, untuk mendapatkan daftar agen dari infrastruktur Wazuh, seorang analis harus berinteraksi dengan API melalui antarmuka baris perintah (CLI) dalam dua tahapan yang kaku. Pertama, analis harus mengirimkan request autentikasi menggunakan kredensial untuk mendapatkan JSON Web Token (JWT). Kedua, analis harus menyisipkan *token* tersebut ke dalam *header* HTTP

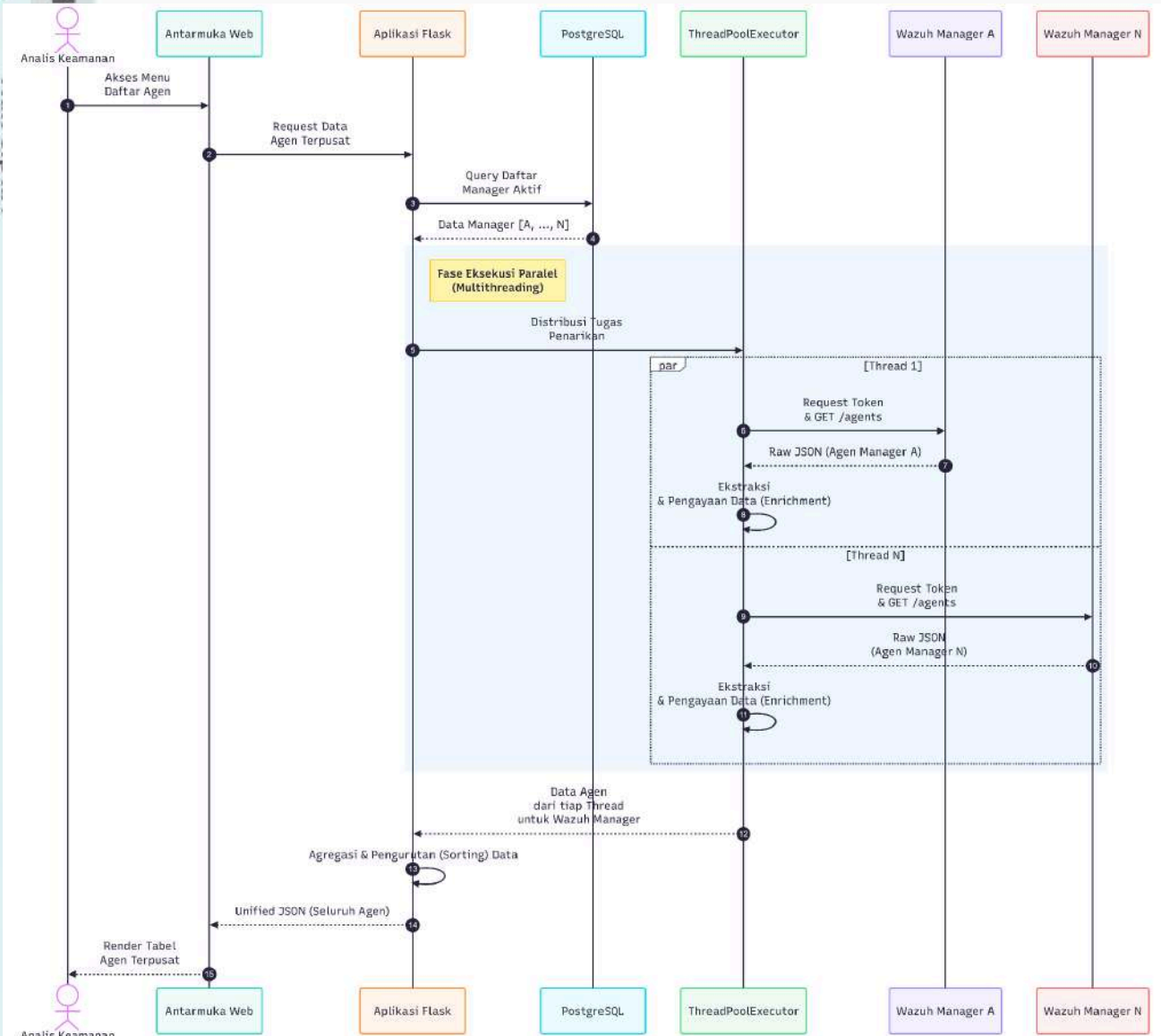
```
chriss@kali:~$ curl -su v [REDACTED] -- -k -X POST "https://[REDACTED]:55000/security/user/authenticate"
```

```
{  
  "data": {  
    "token": "eyJhbGciOiJIJFZlbnRlcCI6Ikpvcj9yYjc3MioiOiJJYXplcCIsImF1ZCI6IldhenVoIEQSSSRVNUiWibmJmIjozNzc4NDUwLmZzczLCJleHAIOjE3NgzwNTYyNmEsInNlbSI6IndhenVolXdlaSisInjb1bi9hcyciOjE6ZWZsc2UsInRpLnRermfDz02ufx6u49c3X9WkaES9Up-52ReDKKixAWque4GA7sl5UX2cpWid2hdpgUiFo.AL2AgtFzyWNdSlajag-oTjsDsJgHqI8ZLOd9WaXHCUkgDH-NtZRrmfDz02ufx6u49c3X9WkaES9Up-52ReDKKixAWque4GA7sl5UX2cpW  
chriss@kali:~$ viZtcDk -X GET "https://[REDACTED].168.111.2:55000/agents" -H "Authorization: Bearer eyJhbGciOiJIJFZlbnRlcCI6Ikpvcj9yYjc3MioiOiJJYXplcCIsImF1ZCI6IldhenVoIEQSSSRVNUiWibmJmIjozNzc4NDUwLmZzczLCJleHAIOjE3NgzwNTYyNmEsInNlbSI6IndhenVolXdlaSisInjb1bi9hcyciOjE6ZWZsc2UsInRpLnRermfDz02ufx6u49c3X9WkaES9Up-52ReDKKixAWque4GA7sl5UX2cpWsqgzmnxQUXPbj4vdtLSrcHpsoD7lz9YIMURPyoNZaend5YghFW7Yq2akFeTRFP9WBXLFAOCOMxiI9_QcuCF7n155vgwtfdHPWSxa_VShJRSAcrPeKg0zd6dqihRLot7ffzAXfe0KFTxHJ6WA7p32KDULHW" | jq  
  
% Total      % Received   Xferd        Average Speed          Time           Time       Time     Current  
             Dload Upload    Total         Spent            Left     Speed  
  
100 2917    100 2917      0       0 49358              0  -:--:-:    -:--:-:    -:--:-: 49440  
{  
  "data": {  
    "affected_items": [  
      {  
        "os": {  
          "arch": "x86_64",  
          "major": "2023",  
          "name": "Amazon Linux",  
          "platform": "amzn",  
          "uname": "Linux |wazuh.manager|5.15.0-161-generic|#171-Ubuntu SMP Sat Oct 11 08:17:01 UTC 2025|x86_64",  
          "version": "2023"  
        },  
        "name": "wazuh.manager",  
        "status": "active",  
        "status_code": 0,  
        "registerIP": "[REDACTED]",  
        "dateAdd": "2026-04-01T03:52:43+00:00",  
        "lastKeepAlive": "9999-12-31T23:59:59+00:00",  
        "node_name": "node01",  
        "id": "000",  
        "manager": "wazuh.manager",  
        "version": "Wazuh v4.14.4",  
        "group config status": "sync'd"  
      }  
    ]  
  }  
}
```

Gambar 4.41 Proses Manual Penarikan Data Agen Wazuh via CLI (Sistem Lama *As-Is*)

Melakukan dua tahap proses tersebut secara manual untuk satu server saja sudah membutuhkan waktu dan interaksi yang tidak sedikit. Pada lingkungan yang memiliki banyak Wazuh Manager, proses pengambilan data secara *sequential* dari setiap server akan semakin meningkatkan latensi. Akibatnya, administrator tidak memperoleh gambaran kondisi seluruh infrastruktur secara cepat, sehingga berpotensi menimbulkan titik buta (*blind spot*) operasional yang dapat menghambat pemantauan keamanan.

Untuk memitigasi *bottleneck* tersebut dan membebaskan analis dari proses administratif, sistem dirancang dengan arsitektur pemrosesan paralel (*multithreading*). Alur komputasi ini dimodelkan dalam diagram sekuensial pada Gambar 4.42.



Gambar 4.42 Diagram Alur Sekuensial Sistem Penarikan Data Multi-Agen secara Paralel (Sistem Proposed)

Berdasarkan diagram di atas, alur kerja sistem dalam melakukan agregasi data diabstraksikan ke dalam tiga tahapan logis:

- Fase Inisiasi: Saat halaman dimuat, *backend* pertama-tama akan mengumpulkan daftar seluruh *Wazuh Manager* yang berstatus aktif dari basis data PostgreSQL (Langkah 3-4).
- Fase Eksekusi Paralel: Sistem mendistribusikan tugas penarikan data ke dalam sebuah *thread pool*. Setiap *thread* akan bekerja secara independen



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

untuk mengautentikasi, menarik data JSON, dan memfilter struktur data dari masing-masing manajer secara bersamaan (Langkah 5-11).

- iii. Fase Agregasi dan Visualisasi: Setelah seluruh *thread* menyelesaikan tugasnya, sistem menggabungkan seluruh respons parsial menjadi satu daftar utuh, mengurutkannya, lalu mengembalikannya ke antarmuka pengguna untuk dirender menjadi tabel terpusat (Langkah 12-15).

B. Eksekusi API Paralel dan Pemrosesan Data

Implementasi teknis dari alur sekuensial di atas dikembangkan menggunakan modul *concurrent.futures.ThreadPoolExecutor* bawaan Python. Pendekatan ini memungkinkan *backend* aplikasi web (Flask) untuk mendelegasikan tugas-tugas yang terhambat oleh proses Input/Output (I/O bound) jaringan ke latar belakang, sehingga alur komputasi utama tetap responsif.

Berikut adalah potongan kode yang mengimplementasikan orkestrasi pemanggilan API secara paralel:

```

1 # Potongan Fungsi Kode Agregasi Paralel
2 from concurrent.futures import ThreadPoolExecutor, as_completed
3
4 def get_and_cache_all_agents(app, force_refresh=False):
5     # 1. Mengambil daftar seluruh Wazuh Manager yang aktif dari basis data
6     active_managers = WazuhManager.query.filter_by(is_active=True).all()
7     all_agents = []
8
9     # 2. Inisiasi ThreadPoolExecutor dengan jumlah pekerja (worker) sesuai total
manajer
10     with ThreadPoolExecutor(max_workers=len(active_managers) or 1) as executor:
11
12         # 3. Mendelegasikan tugas ke masing-masing thread secara bersamaan
(Paralel)
13         future_to_manager = {
14             executor.submit(_get_agents_from_single_manager, app, manager):
manajer
15             for manager in active_managers
16         }
17
18         # 4. Menangkap hasil (JSON) segera setelah sebuah thread selesai
bertugas
19         for future in as_completed(future_to_manager):
20             manager = future_to_manager[future]
21             try:
22                 # 5. Menggabungkan (extend) hasil penarikan dari satu manajer ke

```



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
 a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta

2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

daftar utama
23         agents_from_manager = future.result()
24         all_agents.extend(agents_from_manager)
25     except Exception as exc:
26         app.logger.error(f"Thread error pada manajer '{manager.name}':
{exc}")
27
28     # 6. Mengurutkan daftar akhir berdasarkan nama manajer dan nama agen
29     all_agents.sort(key=lambda x: (x.get("_source_manager_name", ""),
x.get("name", "")))
30
31     return all_agents, "api"

```

Gambar 4.43 Potongan Fungsi Kode Agregasi Paralel

Pada Langkah 4 di atas, fungsi *as_completed()* memainkan peran penting dalam efisiensi komputasi. Fungsi ini tidak menunggu *thread* selesai berdasarkan urutan pengiriman, melainkan langsung memproses *thread* mana pun yang selesai lebih dulu.

Selain eksekusi paralel, sistem juga melakukan proses Pengayaan Data (*Data Enrichment*) di dalam fungsi *_get_agents_from_single_manager*. Karena struktur JSON asli dari API Wazuh tidak mencantumkan dari *server* mana sebuah agen berasal, *backend* secara otomatis menyisipkan atribut tambahan ke dalam setiap objek agen sebelum digabungkan.

```

1 # Potongan Kode Fungsi Pengayaan Data di dalam Thread
2 for agent in raw_agents:
3     # Pengayaan Data: Menambahkan konteks sumber ke dalam respons JSON
mentah
4     agent["_source_manager_name"] = manager.name
5     agent["_source_manager_id"] = manager.id
6     agent["_source_manager_url"] = manager.url
7
8     agents_list.append(agent)

```

Gambar 4.44 Potongan Kode Fungsi Pengayaan Data di dalam *Thread*

Melalui injeksi atribut spesifik seperti *_source_manager_name*, sistem memastikan bahwa saat ratusan agen dari berbagai kluster digabungkan ke dalam satu tabel di antarmuka, analis keamanan tetap dapat mengidentifikasi dan memfilter agen berdasarkan *server* induknya dengan presisi. Sehingga, kini sudah



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

ada *field* tambahan yang datanya tidak diambil dari Wazuh Manager tetapi dari enrichment seperti pada Gambar 4.45

4 Agents Found

ID	STATUS	IP ADDRESS	AGENT NAME	MANAGER NAME	MANAGER URL
002	Disconnected	192.168.111.2	Server	Wazuh Main	https://192.168.111.2:55000
001	Disconnected	192.168.111.3	Server	Wazuh Main	https://192.168.111.2:55000
003	Active	192.168.111.3	Server	Wazuh Main	https://192.168.111.2:55000
000	Active	127.0.0.1	wazuh	Wazuh Main	https://192.168.111.2:55000

Gambar 4.45 *Field Manager Name* yang diambil dari *enrichment*

C. Mekanisme *Caching* Operasional (Redis)

Data hasil penarikan *agent* dari seluruh Wazuh Manager tidak disimpan secara persisten di PostgreSQL seperti data Wazuh Manager dan Wazuh Indexer karena karakteristiknya yang dinamis seiring kondisi tiap *endpoint*. Menyimpan data tersebut secara permanen berpotensi menyebabkan informasi yang ditampilkan menjadi tidak mutakhir. Meskipun demikian, sistem tetap memerlukan mekanisme untuk menghimpun dan menyediakan data tersebut secara terpusat agar dapat diakses administrator. Informasi daftar dan status *agent* dapat disajikan secara efisien tanpa mengorbankan keterbaruan data.

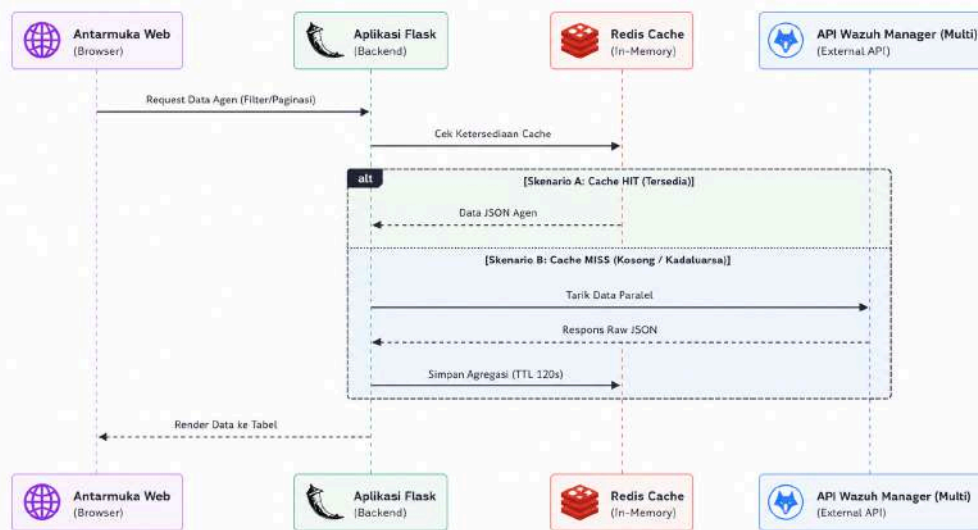
Walaupun arsitektur penarikan data secara paralel telah mempercepat proses komputasi secara signifikan, melakukan pemanggilan API eksternal setiap kali pengguna melakukan pencarian atau berpindah halaman tetap berpotensi meningkatkan beban jaringan dan waktu respons.

Oleh karena itu, sistem mengimplementasikan lapisan *caching* operasional menggunakan Redis sebagai tempat penyimpanan sementara sekaligus penghimpunan data *agent*. Mekanisme ini memungkinkan data disajikan dengan *load time* yang lebih cepat tanpa harus terus-menerus melakukan permintaan ke setiap server. Alur pengambilan keputusan sistem dalam memanfaatkan memori Redis diabstraksikan pada diagram Gambar 4.46 sebagai abstraksi mekanisme *caching* operasional.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 4.46 Diagram Abstraksi Mekanisme *Caching* Operasional

Hasil agregasi dari seluruh agen yang ditarik pada tahap sebelumnya tidak langsung dibuang setelah dirender, melainkan dikonversi menjadi format *string* JSON dan disimpan di dalam RAM Redis.

Berikut adalah implementasi logika *caching* pada proses penarikan data agen:

```

1 Potongan Kode Logika Redis Caching
2 def get_and_cache_all_agents(app, force_refresh=False):
3     redis_client = app.redis
4     REDIS_AGENTS_KEY = "sys:wazuh:agents_data"
5     REDIS_AGENTS_TTL = 120 # Batas waktu kadaluarsa: 2 menit
6
7     # 1. Cek Cache: Jika tidak dipaksa refresh, cari data di memori
8     if not force_refresh and redis_client:
9         try:
10             cached_data = redis_client.get(REDIS_AGENTS_KEY)
11             if cached_data:
12                 # 2. Cache HIT: Langsung kembalikan data dari RAM
13                 return json.loads(cached_data), "cache"
14             except Exception as e:
15                 app.logger.error(f"Gagal membaca cache Redis: {e}")
16
17     # 3. Cache MISS / Force Refresh: Lakukan penarikan API paralel
18     (Logika sebelumnya)
  
```



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

18 # ... [Proses ThreadPoolExecutor berjalan di sini] ...
19
20 # 4. Simpan Cache: Menyimpan hasil agregasi ke Redis dengan TTL
120 detik
21 if redis_client:
22     try:
23         redis_client.setex(REDIS_AGENTS_KEY, REDIS_AGENTS_TTL,
24                             json.dumps(all_agents))
25     except Exception as e:
26         app.logger.error(f"Gagal menyimpan cache ke Redis: {e}")

```

Gambar 4.47 Potongan Kode Logika Redis *Caching*

Berdasarkan implementasi pada kode, data agen disimpan dengan batas waktu (Time-To-Live / TTL) selama 120 detik (2 menit). Apabila Administrator memerlukan data terbaru secara langsung, misalnya saat melakukan *troubleshooting agent*, sistem menyediakan tombol Refresh Now yang mengirimkan parameter *force_refresh=True* (Langkah 1) untuk melewati *cache* Redis dan mengambil data terbaru langsung dari API Wazuh.

D. Antarmuka Visualisasi dan Fitur *Filtering*

Data agregasi multi-agen yang telah dikumpulkan divisualisasikan melalui antarmuka pengguna berbasis *Single Pane of Glass*. Antarmuka ini membebaskan analis dari keharusan untuk *login* ke berbagai *dashboard* manajer yang berbeda hanya untuk memantau status aset organisasi.

ID	STATUS	IP ADDRESS	AGENT NAME	MANAGER NAME	MANAGER URL
002	Disconnected	192.168.111.2	Server	Wazuh Main	https://192.168.111.2:55000
001	Disconnected	192.168.111.3	Server	Wazuh Main	https://192.168.111.2:55000
003	Active	192.168.111.3	Server	Wazuh Main	https://192.168.111.2:55000
000	Active	127.0.0.1	wazuh	Wazuh Main	https://192.168.111.2:55000

Gambar 4.48 Tampilan UI Wazuh *Agents*



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Seperti yang ditunjukkan pada Gambar 4.48, setiap agen direpresentasikan dalam baris tabel yang dilengkapi dengan indikator status visual (hijau untuk Active, merah untuk Disconnected, kuning untuk Pending, dan abu-abu untuk Never Connected).

Untuk manajemen data agen yang berjumlah masif, antarmuka dilengkapi dengan sistem *filtering* dan pencarian hibrida. Implementasi pencarian pada *frontend* (JavaScript) dibangun menggunakan teknik *debouncing*. Teknik ini secara sengaja menunda pengiriman request ke *backend* hingga pengguna berhenti mengetik selama 400 milidetik, sehingga mencegah terjadinya *spam* request yang dapat membuat *server* kelebihan beban.

ID	STATUS	IP ADDRESS	AGENT NAME	MANAGER NAME	MANAGER URL
091	Disconnected	192.168.111.3	Server-RISK	Wazuh Main	https://192.168.111.2:55609
092	Active	192.168.111.3	Server-Risk	Wazuh Main	https://192.168.111.2:55609

Gambar 4.49 Tampilan Pencarian dan *Filtering Agent*

Selain itu, sistem mengimplementasikan pembaruan status URL (URL *State Management*) secara dinamis seperti pada Gambar 4.50.

```

1 // Potongan Kode JS Logika URL State Management
2 function updateURLState() {
3   // 1. Membaca parameter pencarian dan filter aktif dari state
  aplikasi
4   const params = new URLSearchParams();
5   if (state.currentPage > 1) params.set('page', state.currentPage);
6   if (state.managerId) params.set('manager_id', state.managerId);
7   if (state.status) params.set('status', state.status);
8   if (state.searchQuery) params.set('search_query',
state.searchQuery);
9
10  // 2. Memperbarui URL browser tanpa melakukan reload halaman (SPA
behavior)
11  const newUrl =
` ${window.location.pathname}?${params.toString()}`;

```



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```
12     window.history.replaceState({ path: newUrl }, '', newUrl);
13 }
```

Gambar 4.50 Potongan Kode JS Logika URL *State* Management

Melalui fungsi *history.replaceState()* pada Langkah 2, setiap parameter *filter* (seperti *filter* berdasarkan Manajer tertentu atau status *Disconnected*) akan direfleksikan langsung ke dalam URL browser. Hal ini memungkinkan analisis keamanan untuk menyimpan tampilan spesifik atau membagikan tautan URL tersebut kepada anggota tim lain untuk tujuan investigasi kolaboratif.

Terakhir, antarmuka ini dirancang agar saling terhubung dengan modul analisis utama. Nama agen pada tabel dirender sebagai tautan interaktif (*clickable link*).

Agents Found

ID	STATUS	IP ADDRESS	AGENT NAME	MANAGER NAME	MANAGER URL
002	Disconnected	192.168.111.2	Server- [REDACTED]	Wazuh Main	https://192.168.111.2:55000
001	Disconnected	192.168.111.3	Server-RISK	Wazuh Main	https://192.168.111.2:55000
003	Active	192.168.111.3	Server-Risk	Wazuh Main	https://192.168.111.2:55000
008	Active	127.0.0.1	wazuh- [REDACTED]	Wazuh Main	https://192.168.111.2:55000

Gambar 4.51 Tautan Interaktif Agen menuju *Dashboard* Analisis

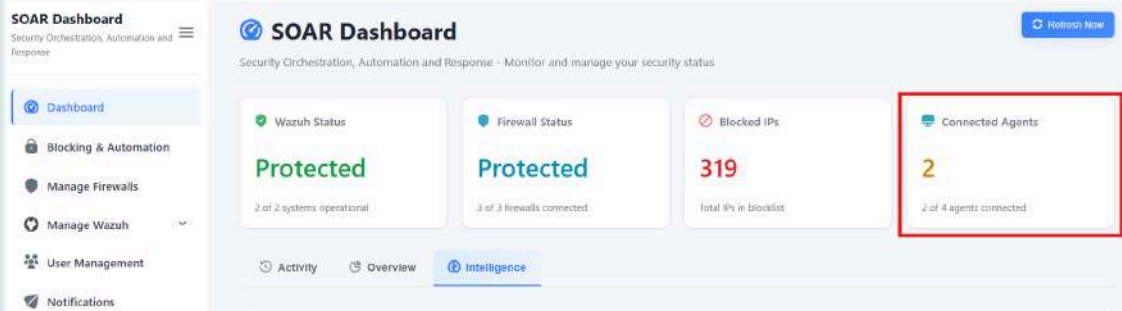
Ketika nama agen diklik, sistem akan langsung mengalihkan pengguna ke halaman *Dashboard* utama dengan membawa parameter nama agen tersebut. Hal ini menciptakan alur kerja investigasi insiden yang sangat instan, di mana analis dapat beralih dari memantau status operasional agen menuju analisis log ancaman keamanan hanya dengan satu klik.

Selanjutnya, jumlah *agents* yang tersambung juga ditunjukkan pada *dashboard* utama untuk menghighlight secara cepat berapa total *agents* yang tersambung. Mekanisme tautan interaktif dipakai kembali untuk membawa pengguna dari *dashboard* tab menuju Wazuh Agents tab secara cepat.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 4.52 Panel Interaktif *Connected Agents* di *Dashboard*

4.3.4 Implementasi Pengayaan Intelijen Ancaman (*Threat Intelligence*)

Untuk mempercepat pengambilan keputusan dalam menangani insiden keamanan, sistem dilengkapi dengan modul Pengayaan Intelijen Ancaman (*Threat Intelligence Enrichment*). Modul ini terintegrasi dengan platform keamanan berbasis komunitas, AlienVault Open Threat Exchange (OTX), yang menyediakan kumpulan data reputasi global terhadap berbagai indikator ancaman siber.

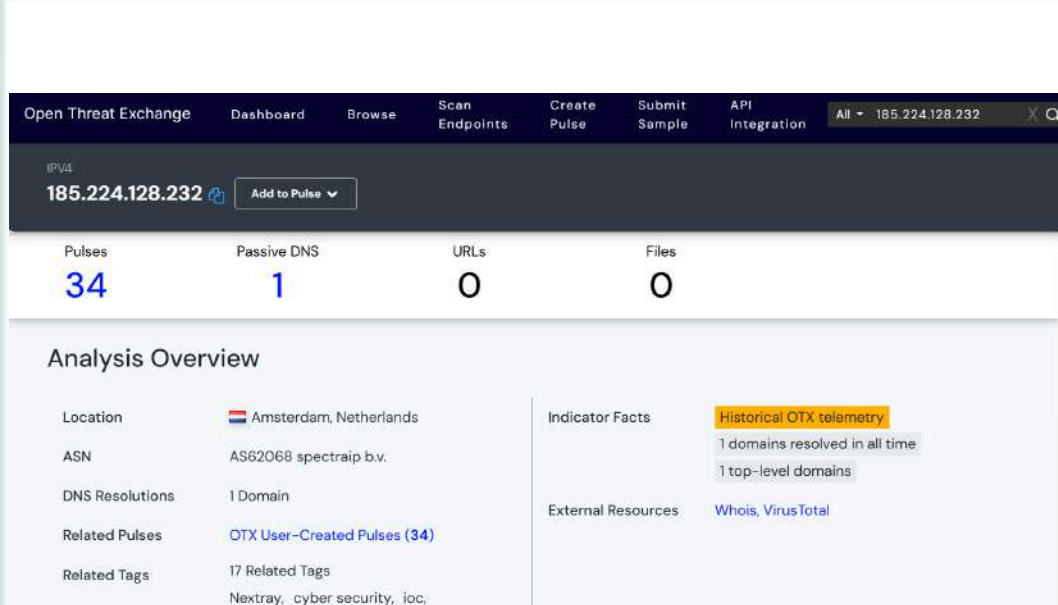
Melalui integrasi ini, informasi reputasi suatu indikator dapat diakses secara langsung dari dalam sistem tanpa perlu membuka layanan eksternal. Analis dapat segera memperoleh konteks tambahan, seperti tingkat reputasi dan informasi ancaman yang terkait, sehingga proses penilaian awal terhadap suatu indikator menjadi lebih cepat dan efisien. Dengan tersedianya informasi tersebut dalam satu antarmuka, investigasi dapat dilakukan dengan alur kerja yang lebih sederhana dan minim perpindahan konteks.

Sebelum integrasi dengan AlienVault OTX diterapkan, proses validasi *Indicator of Compromise* (IoC) dilakukan dengan membuka tab *browser* baru dan menelusuri indikator melalui portal eksternal. Ketika sebuah alamat IP atau *hash* berkas mencurigakan muncul pada log peringatan, analis harus menyalin indikator tersebut, berpindah ke halaman lain, lalu melakukan pencarian secara manual. Rangkaian aktivitas ini menyebabkan *context switching*, yang memperlambat proses investigasi sekaligus meningkatkan jumlah interaksi yang harus dilakukan oleh analis.



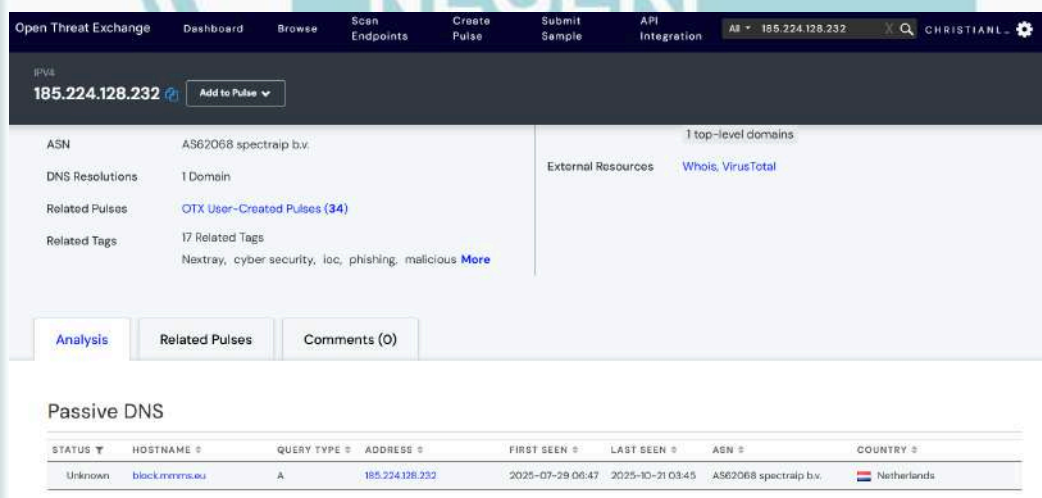
Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 4.53 Antarmuka Web *AlienVault* OTX untuk Mencari Satu IP

Lebih jauh lagi, platform web AlienVault OTX menyajikan hasil analisis dalam beberapa tab yang terpisah, seperti *General*, *Pulses*, *Passive DNS*, dan *Malware*. Akibatnya, analis harus menavigasi beberapa halaman untuk mengumpulkan informasi yang berkaitan sebelum dapat memahami konteks ancaman secara menyeluruh. Proses ini menambah jumlah interaksi yang diperlukan dan memperpanjang waktu investigasi, terutama ketika setiap tab perlu diperiksa. Kondisi tersebut diilustrasikan pada Gambar 4.54.



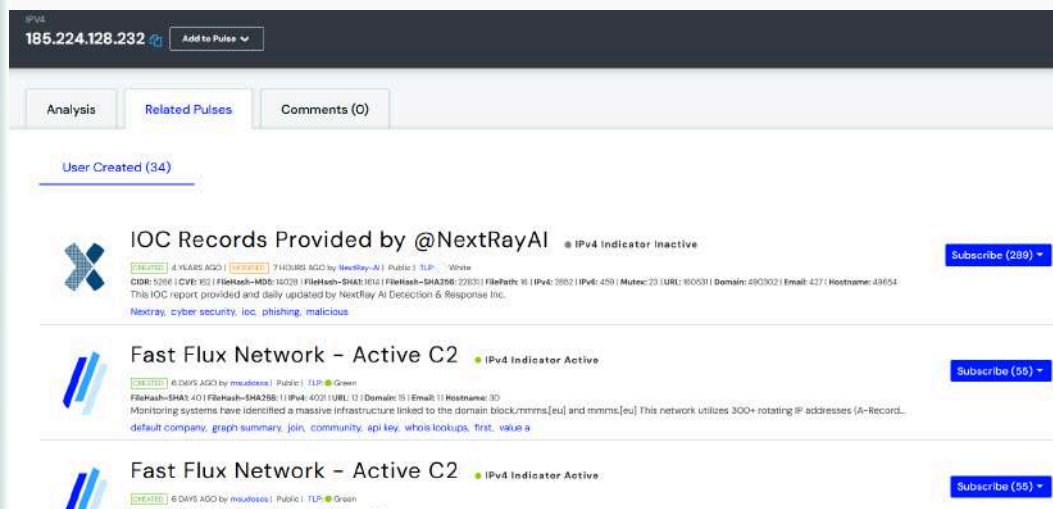
Gambar 4.54 Antarmuka Tab *Analysis* di Web *AlienVault* OTX



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Tampilan ketika melakukan perpindahan tab untuk melakukan pengecekan *Pulses* ditunjukkan seperti pada gambar 4.55.



Gambar 4.55 Antarmuka Tab *Pulses* di Web AlienVault OTX

Sebagai alternatif, analis dapat menggunakan metode baris perintah (CLI) untuk menarik data dengan lebih cepat. Namun, pendekatan ini mengharuskan analis untuk menyusun *header* autentikasi secara manual pada setiap permintaan, seperti yang ditunjukkan pada perintah curl di gambar 4.56. Meskipun lebih cepat, respons yang dikembalikan oleh *server* berupa objek JSON mentah yang sangat padat, bertingkat (*nested*), dan tidak terformat (*unformatted*).

```

chris@chris-laptop:~$ curl -X GET "https://otx.alienvault.com/api/v1/indicators/IPv4/185.224.128.232/general"
-H "X-OTX-API-KEY: [REDACTED]"
{"whois": "http://whois.domaintools.com/185.224.128.232", "reputation": 0, "indicator": "185.224.128.232", "type": "IP
v4", "type_title": "IPv4", "base_indicator": {"id": 3621013801, "indicator": "185.224.128.232", "type": "IPv4", "title
": "", "description": "", "content": "", "access_type": "public", "access_reason": ""}, "pulse_info": {"count": 34, "p
ulses": [{"id": "63456c2a30b92337ea1670e0", "name": "IOC Records Provided by @NextRayAI", "description": "This IOC rep
ort provided and daily updated by NextRay AI Detection & Response Inc.", "modified": "2026-05-07T00:06:47.782000", "cr
eated": "2022-10-11T13:14:18.676000", "tags": ["NextRay", "cyber security", "ioc", "phishing", "malicious"], "referenc
es": [], "public": 1, "adversary": "", "targeted_countries": ["United States of America", "Turkey", "Ukraine", "Romani
a", "Czechia", "United Kingdom of Great Britain and Northern Ireland", "Norway", "Lithuania", "Estonia", "Latvia", "Po
land", "Germany", "Canada", "France", "Denmark"], "malware_families": [], "attack_ids": [], "industries": ["Defense",
"Industrial", "Government"], "TLP": "White", "cloned_from": null, "export_count": 1326, "upvotes_count": 0, "downvotes
_count": 0, "votes_count": 0, "locked": false, "pulse_source": "api", "validator_count": 0, "comment_count": 0, "follo
wer_count": 0, "vote": 0, "author": {"username": "NextRay-AI", "id": "210822", "avatar_url": "/otxapi/users/avatar_ima
ge/media/avatars/user_210822/resized/80/avatar_3b9c358f36.png", "is_subscribed": false, "is_following": false}, "indic
ator_type_counts": {"domain": 490302, "IPv4": 2852, "IPv6": 459, "hostname": 49654, "URL": 160531, "CIDR": 5266, "File
Hash-MD5": 14028, "FileHash-SHA256": 22831, "CVE": 162, "email": 427, "Mutex": 23, "FileHash-SHA1": 1614, "FilePath":
16}, "indicator_count": 748165, "is_author": false, "is_subscribing": null, "subscriber_count": 289, "modified_text":
"6 hours ago", "is_modified": true, "groups": [], "in_group": false, "threat_hunter_scannable": true, "threat_hunter
_has_agents": 1, "related_indicator_type": "IPv4", "related_indicator_is_active": 0}, {"id": "69f44c4688a318071a056f09"
, "name": "Fast Flux Network - Active C2", "description": "Monitoring systems have identified a massive infrastructure
linked to the domain block.mnms.eu and mnms.eu This network utilizes 300+ rotating IP addresses (A-Records) to ma
intain persistence. This behavior is consistent with high-level botnet Command & Control (C2) activity, potentially li
nked to malware delivery (e.g., Mirai, QakBot).2. Technical DetailsTarget Domain: mnms.eu / network.block.mnms.euInfra
structure Pattern: Fast-Flux DNS (IPs rotate every 59 seconds).Hosting Providers: High density across DigitalOcean, AW
S, Linode, and various offshore VPS providers. The classification as \"Vehicles\" on alphaMountain.ai is a significant

```

Gambar 4.56 Respons Mentah (*Raw JSON*) dari API *AlienVault* OTX via Terminal



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Proses ekstraksi informasi ancaman secara manual tidak hanya menambah jumlah langkah yang harus dilakukan analis, tetapi juga meningkatkan beban kognitif selama investigasi.

Setelah memperoleh informasi dari platform *Threat Intelligence*, analis masih harus mencatat dan memindahkan hasil temuannya kembali ke sistem pelaporan internal. Rangkaian aktivitas tersebut mengkonsumsi waktu yang tidak sedikit dan dapat memperlambat proses penanganan insiden, sehingga organisasi semakin sulit mencapai *breakout time* yang ideal. Secara umum, beban tersebut meliputi:

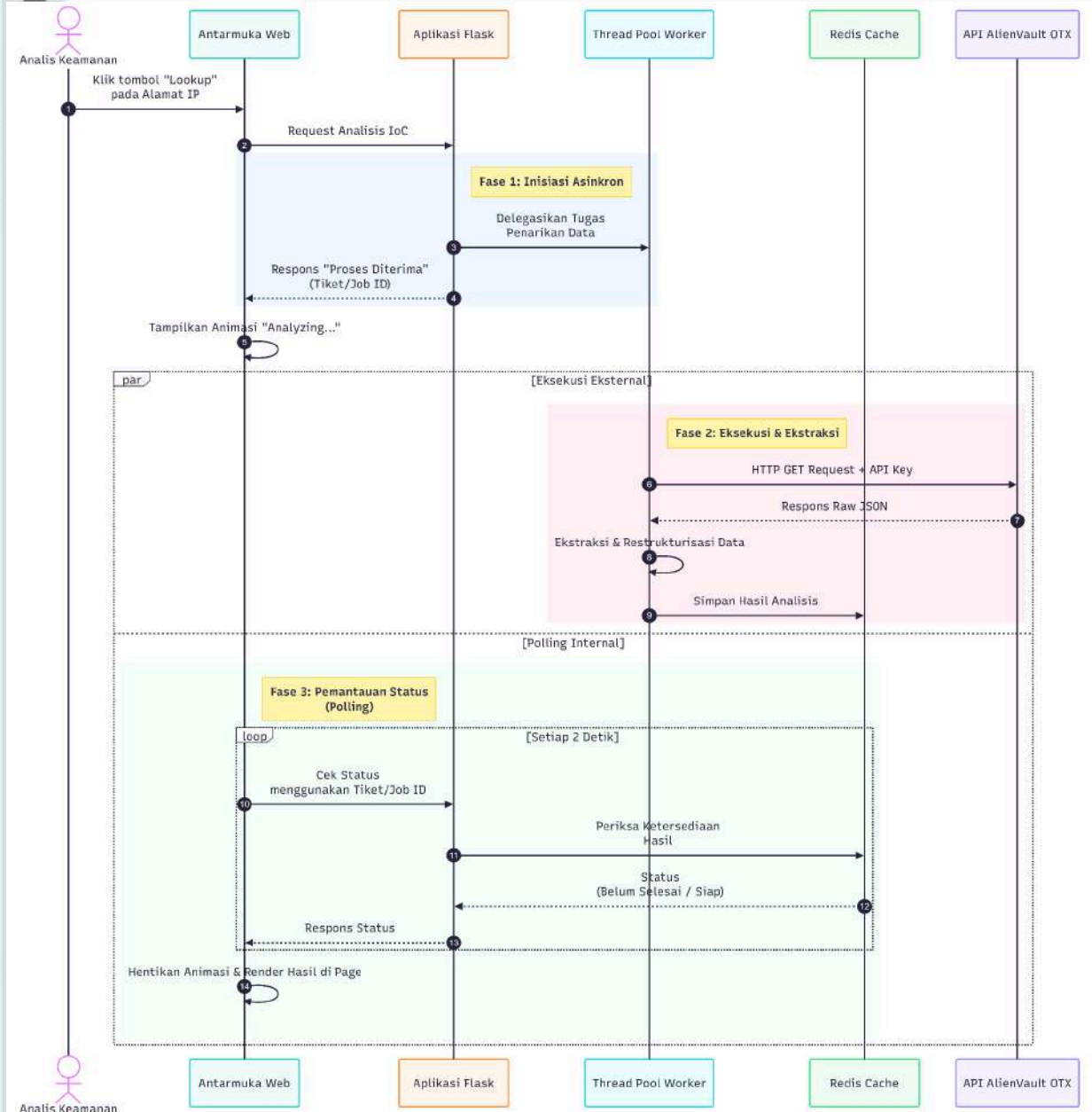
- a) Melakukan perpindahan antar aplikasi atau tab (*context switching*).
- b) Menyalin dan menempel (*copy-paste*) indikator serta hasil investigasi secara manual.
- c) Menafsirkan dan merangkum informasi ancaman dari sumber eksternal.
- d) Mendokumentasikan kembali hasil investigasi ke dalam sistem pelaporan internal.

Untuk mengeliminasi rangkaian aktivitas yang berulang tersebut, sistem mengabstraksikan seluruh proses investigasi ke dalam satu tombol pintasan (Lookup) pada antarmuka. Ketika tombol dipilih, sistem secara otomatis mengirimkan indikator ke layanan AlienVault OTX, mengambil informasi reputasi, dan menampilkan hasilnya tanpa memerlukan interaksi tambahan dari pengguna.

Karena proses pengambilan data dari layanan eksternal dapat memerlukan beberapa detik, mekanisme ini diimplementasikan menggunakan pola Investigasi Asinkron. Pendekatan tersebut memastikan antarmuka pengguna tetap responsif selama proses berlangsung, sehingga pengguna dapat tetap berinteraksi dengan aplikasi sambil menunggu hasil investigasi. Alur komputasi ini dimodelkan pada diagram sekuensial pada Gambar 4.57.



CH



Gambar 4.57 Diagram Alur Sekuensial Investigasi Asinkron (*Threat Intelligence*)

Berdasarkan diagram di atas, kompleksitas penarikan data dari API eksternal yang memakan waktu diabstraksikan ke dalam tiga fase utama yang berjalan secara asinkron:

- i. Fase Inisiasi Asinkron (Langkah 2-4): Saat analis menekan tombol *Lookup*, sistem tidak membuat antarmuka web menunggu hingga proses investigasi selesai. Sebaliknya, *backend* segera mendelegasikan tugas pengambilan data ke Pekerja Latar Belakang (*Thread Pool*) dan langsung

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

mengembalikan sebuah *Job ID* sebagai identitas proses investigasi yang sedang berlangsung. Pendekatan ini membuat permintaan pengguna dapat diproses tanpa menghambat respons aplikasi, sehingga antarmuka tetap responsif dan analis dapat melanjutkan aktivitas lain sambil menunggu hasil investigasi.

- ii. Fase Eksekusi & Ekstraksi (Langkah 6-9): Setelah menerima tugas, *thread* yang berjalan secara independen menghubungi API AlienVault OTX dengan menyertakan *API Key* sebagai mekanisme autentikasi. Data hasil investigasi kemudian diterima dalam format JSON dan diproses lebih lanjut untuk mengekstraksi informasi yang relevan, seperti reputasi indikator dan detail ancaman. Setelah proses tersebut selesai, hasil yang telah dirapikan disimpan sementara di Redis agar dapat diakses dengan cepat oleh antarmuka web tanpa perlu melakukan permintaan ulang ke layanan eksternal.
- iii. Fase Pemantauan Status / *Polling* (Langkah 10-15): Selama proses investigasi berlangsung di latar belakang, antarmuka web menggunakan *Job ID* untuk memantau status pekerjaan secara berkala setiap dua detik. Pada setiap permintaan, sistem hanya memeriksa apakah hasil investigasi untuk *Job ID* tersebut telah tersedia di Redis. Setelah data ditemukan, proses *polling* dihentikan secara otomatis, animasi *loading* diakhiri, dan ringkasan informasi *threat intelligence* segera ditampilkan kepada analis. Mekanisme ini memberikan pengalaman penggunaan yang lebih lancar karena pengguna tidak perlu menyegarkan halaman atau mengirimkan permintaan secara manual untuk memperoleh hasil investigasi.

B. Implementasi Konektor Eksternal *AlienVault* OTX

Inti dari fase eksekusi latar belakang terletak pada konektor API. Implementasi ini dikembangkan menggunakan bahasa Python untuk mengabstraksikan kerumitan penyusunan *header* dan penanganan *error* jaringan yang sebelumnya harus dilakukan analis secara manual via CLI. Potongan kode implementasi fungsi konektor untuk menarik data intelijen ancaman terjabarkan dalam Gambar 4.58.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

1 # Potongan Kode Fungsi Konektor dan Penarikan Data OTX
2 def fetch_otx_indicator_details(indicator_type, indicator_value,
  api_key, logger):
3     # 1. Abstraksi Autentikasi: Menyisipkan API Key ke dalam header
  khusus OTX secara otomatis
4     headers = {"X-OTX-API-KEY": api_key} if api_key else {}
5
6     # 2. Routing Dinamis: Konstruksi URL berdasarkan tipe IoC (IPv4,
  Domain, atau Hash)
7     general_url =
  f"https://otx.alienvault.com/api/v1/indicators/{indicator_type}/{indicat
  or_value}/general"
8
9     try:
10        # 3. Eksekusi Jaringan: Melakukan HTTP GET request dengan
  batas waktu 30 detik
11        response = requests.get(general_url, headers=headers,
  timeout=30, verify=True)
12        response.raise_for_status()
13        general_data = response.json()
14
15        # 4. Pengayaan Data (Enrichment): Menambahkan atribut visual
  (contoh: URL Bendera Negara)
16        general_data['indicator_type'] = indicator_type
17        if general_data.get('flag_url'):
18            general_data['flag_url_full'] =
  "https://otx.alienvault.com" + general_data['flag_url']
19
20        return general_data
21
22    except requests.exceptions.HTTPError as e:
23        # 5. Penanganan Error Spesifik: 404 bukan berarti aplikasi
  gagal, melainkan IoC aman
24        if e.response.status_code == 404:
25            logger.warning(f>Data tidak ditemukan di database OTX
  untuk indikator '{indicator_value}'.")
26            return {'indicator': indicator_value, 'type':
  indicator_type, 'pulse_info': {}}
27
28        elif e.response.status_code in (401, 403):
29            raise Exception("API Key OTX tidak valid atau tidak
  memiliki otorisasi.")
30
31        raise Exception(f"Gagal mengambil data intelijen siber:
  {e}")

```

Gambar 4.58 Potongan Kode Fungsi Konektor dan Penarikan Data OTX

Kelebihan dari arsitektur fungsi di Gambar 4.58 terlihat pada Langkah 5 (Penanganan *Error Spesifik*). Dalam konteks intelijen ancaman, respons 404 Not



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Found dari *server* OTX bukanlah sebuah kesalahan jaringan (*network error*), melainkan sebuah temuan analitik yang menyatakan bahwa indikator tersebut belum dikenali sebagai ancaman *global* (berpotensi *benign/aman*).

Alih-alih membuat aplikasi *crash* dan memunculkan notifikasi *error* merah di layar analis, sistem menangkap eksepsi tersebut secara anggun (*graceful handling*) dan mengembalikan struktur *dictionary* kosong. Hal ini memungkinkan antarmuka pengguna di Fase 3 untuk tetap menampilkan kartu profil ancaman dengan keterangan "*None detected*", yang memberikan kepastian informasi bagi analis SOC.

C. Alur Penyimpanan dan Keamanan Kredensial OTX

Agar *server backend* dapat melakukan pemanggilan ke API *AlienVault* OTX secara terautentikasi (yang memberikan akses ke rate-limit lebih tinggi dan data yang lebih kaya), Administrator harus mendaftarkan kunci API (*API Key*) ke dalam sistem. Karena *API Key* bersifat rahasia dan rentan disalahgunakan jika bocor, maka sistem harus kembali mengimplementasikan prinsip *Security by Design*.

Data *API Key* dilarang keras disimpan dalam bentuk *plaintext*. Serupa dengan manajemen kredensial infrastruktur Wazuh, sistem menerapkan enkripsi simetris menggunakan algoritma AES pada *layer* model basis data (*SQLAlchemy*) menggunakan fitur *@hybrid_property*. Potongan kode yang mengilustrasikan mekanisme enkripsi tersebut terlihat pada Gambar 4.59.

```

1 # Potongan Model Penyimpanan Threat Intelligence
2 from cryptography.fernet import Fernet
3 from sqlalchemy.ext.hybrid import hybrid_property
4
5 class ThreatIntelligenceProvider(db.Model):
6     # 1. Field fisik di database yang menyimpan ciphertext dari API
7     _encrypted_api_key = db.Column("encrypted_api_key",
8                                     db.LargeBinary, nullable=True)
9
10    @hybrid_property
11    def api_key(self):
12        # 2. Getter: Mendekripsi data biner menjadi teks kembali
13        hanya saat diakses memori

```



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

12         if not self._encrypted_api_key: return None
13     try:
14         return
15     self._get_crypter().decrypt(self._encrypted_api_key).decode()
16     except Exception:
17         current_app.logger.error("Gagal mendekripsi API Key
18         OTX.")
19         return None
20
21     @api_key.setter
22     def api_key(self, plaintext_api_key):
23         # 3. Setter: Mengenkripsi input teks terang menjadi format
24         biner saat disimpan
25         if not plaintext_api_key:
26             self._encrypted_api_key = None
27         else:
28             self._encrypted_api_key = \
29             self._get_crypter().encrypt(plaintext_api_key.encode())

```

Gambar 4.59 Potongan Model Penyimpanan *Threat Intelligence*

Melalui *setter* pada Langkah 3 di kode Gambar 4.59, setiap kali Administrator memasukkan API Key baru melalui formulir konfigurasi, sistem secara transparan akan mengubah *string* tersebut menjadi *ciphertext* sebelum menyimpannya ke PostgreSQL. Pendekatan ini memastikan bahwa bahkan jika peretas (*attacker*) berhasil mengakses *dump* basis data secara penuh, kredensial integrasi layanan pihak ketiga akan tetap terlindungi.

Bentuk dari *table threat_intelligence_providers* yang berisi *field id, name* dan *encrypted_api_Key* yang sudah berbentuk *plaintext* dapat terlihat seperti pada *snippet database* Gambar 4.60.

id	name	encrypted_api_key	is_active
1	AlienVault OTX	gAAAAABo9gQjmIm8F9hVzJA7AzGUGicOJJ0dFzy3h31LIInoru58Jvz7BzuKOjd4pX8cZI	true

Gambar 4.60 Bentuk *Ciphertext* dari *encrypted_api_key*

D. Antarmuka Pemetaan Konteks Ancaman dan JSON Explorer

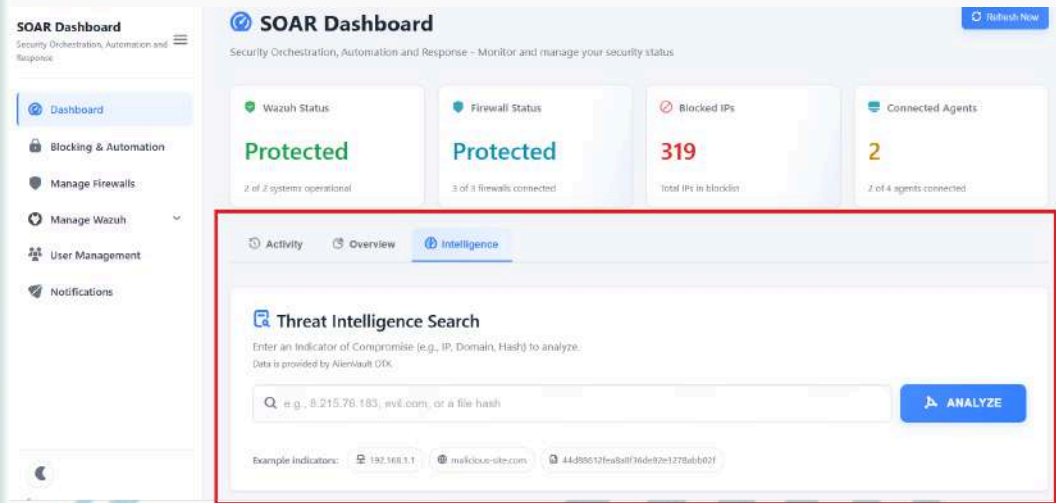
Hasil akhir dari seluruh proses asinkron dan ekstraksi data di *backend* disajikan melalui *dashboard* interaktif pada tab Intelligence. Antarmuka ini dirancang untuk meminimalkan beban kognitif analis dengan menyajikan



Hak Cipta :

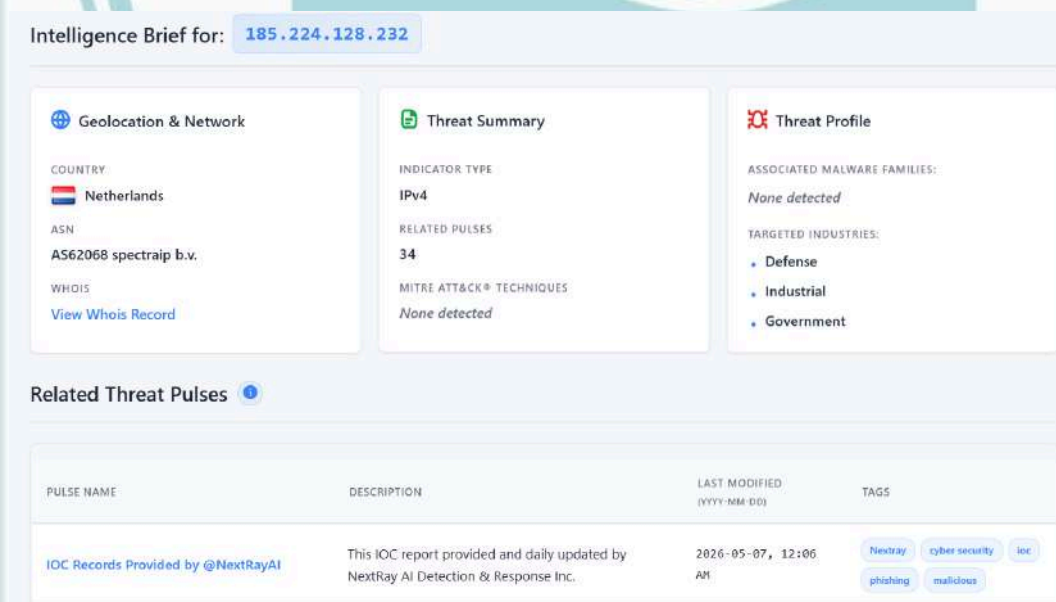
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

informasi forensik yang kompleks ke dalam struktur visual yang intuitif dan mudah dipahami.



Gambar 4.61 Bagian Tab Intelligence di *Dashboard*

Antarmuka pencarian dibangun untuk menjadi tempat pencarian yang dinamis, bukan hanya untuk mencari IP. Analis dapat memasukkan berbagai tipe *Indicator of Compromise* (IoC) ke dalam satu kolom pencarian yang sama (mulai dari alamat IPv4, nama domain, hingga nilai *hash* MD5/SHA256).



Gambar 4.62 Hasil *Threat Intelligence* di Modul untuk Pencarian IoC IPv4

Backend (sebagaimana diuraikan pada fungsi konektor di Bagian B) akan mendeteksi format input secara otomatis dan mengarahkannya ke *endpoint* API



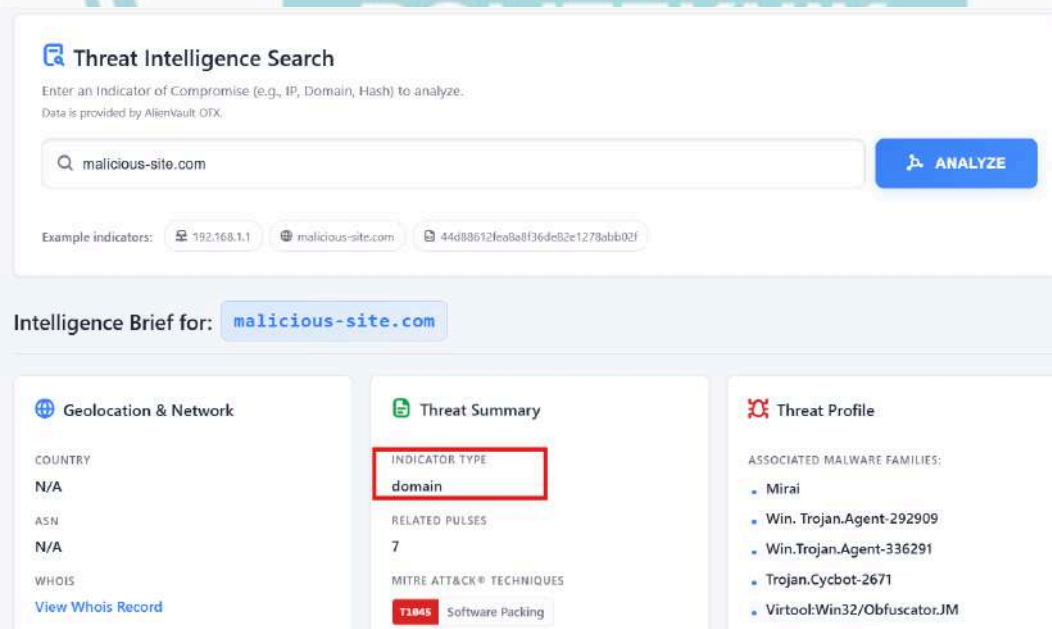
Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

OTX yang tepat. Setelah proses *polling* asinkron selesai, sistem akan merender hasil analisis (*Intelligence Brief*) ke dalam layar tanpa melakukan reload halaman.

Seperti yang diilustrasikan pada Gambar 4.63, sistem akan menampilkan secara otomatis *Indicator Type* yang sesuai dengan apa yang dimasukkan user. Selain itu, sistem akan menerjemahkan JSON mentah dari OTX ke dalam tiga segmen visual utama:

1. *Geolocation & Network*: Menampilkan informasi infrastruktur jaringan dari alamat IP, diperkaya dengan ikon bendera negara (*Data Enrichment*) agar analis dapat dengan cepat mengenali asal geografis serangan.
2. *Threat Summary*: Menampilkan metrik utama dari IoC tersebut, termasuk taktik dan teknik serangan yang dipetakan secara langsung dengan kerangka kerja industri standar, yaitu MITRE ATT&CK®. Setiap nama teknik dirender sebagai label (*pill*) berwarna merah untuk mempertegas tingkat bahaya.
3. *Threat Profile*: Mengagregasi data dari seluruh kampanye serangan (*Threat Pulses*) yang terkait, lalu merangkum jenis keluarga *malware* (misal: SSH *Brute-Force*, Mirai) dan industri yang menjadi target serangan utama.



Gambar 4.63 Hasil *Threat Intelligence* di Modul untuk Pencarian IoC domain



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Selain menyajikan indikator reputasi dasar, sistem juga dirancang untuk mengabstraksi data kontekstual yang dikenal sebagai *Threat Pulses*. Dalam ekosistem *AlienVault OTX*, sebuah *Pulse* merupakan laporan intelijen berbasis komunitas yang mengelompokkan berbagai Indikator Kompromi (IoC) ke dalam satu kampanye serangan spesifik, kelompok *Threat Actor*, atau jenis *malware* tertentu.

Alih-alih membiarkan analis mencari manual atau memilah struktur JSON yang rumit, *backend* sistem secara otomatis mengekstraksi atribut-atribut krusial dari *array Pulse* tersebut dan merendernya ke dalam antarmuka *tabular* yang bersih. Seperti yang diilustrasikan pada Gambar 4.64, sistem menonjolkan empat elemen utama: Nama Kampanye (*Pulse Name*), Deskripsi taktis serangan, Waktu modifikasi terakhir, serta sekumpulan label (*Tags*).

Related Threat Pulses ⓘ

PULSE NAME	DESCRIPTION	LAST MODIFIED (YYYY-MM-DD)	TAGS
Dropbox Typo squatting campaign. CoolW...	Dropbox Typo squatting campaign. Mirai and Ransomware Many domains affected. Has been executed. Threat actor attacked bank/s and Dropbox via Drive by compromises and malicious redirects. Multiple Dropbox accounts added to customer accounts confuse bank and customers. All accounts kept until Bank experienced serious breach. Bank admits to breach. Unsure if made public. Customer suddenly loses all paid storage, business tools, registered domains, and investment accounts. Bank empathizes targeted attacks. Occurred post initial infection & Pegasus Attack by same threat actors.	2026-04-23, 04:01 AM	domain ipv4 ck t1045 run keys startup web protocols tool transfer user execution dns accept +110 more
Dropbox Typo squatting campaign. CoolW...	Dropbox Typo squatting campaign. Mirai and Ransomware Many domains affected. Has been executed. Threat actor attacked bank/s and Dropbox via Drive by compromises and malicious redirects. Multiple Dropbox accounts added to customer accounts confuse bank and customers. All accounts kept until Bank experienced serious breach. Bank admits to breach. Unsure if made public. Customer suddenly loses all paid storage, business tools,	2026-04-23, 04:01 AM	domain ipv4 ck t1045 run keys startup web protocols tool transfer user execution dns accept +110 more

Gambar 4.64 Tampilan Related *Threat Pulses*

Selain rangkuman visual, sistem juga mengakomodasi kebutuhan investigasi lebih mendalam (untuk *deep-dive forensic*) dengan menyediakan komponen JSON Explorer.

Komponen JSON Explorer (Gambar 4.65) memungkinkan analis *senior* untuk menelusuri seluruh data mentah (*raw data*) yang dikembalikan oleh API

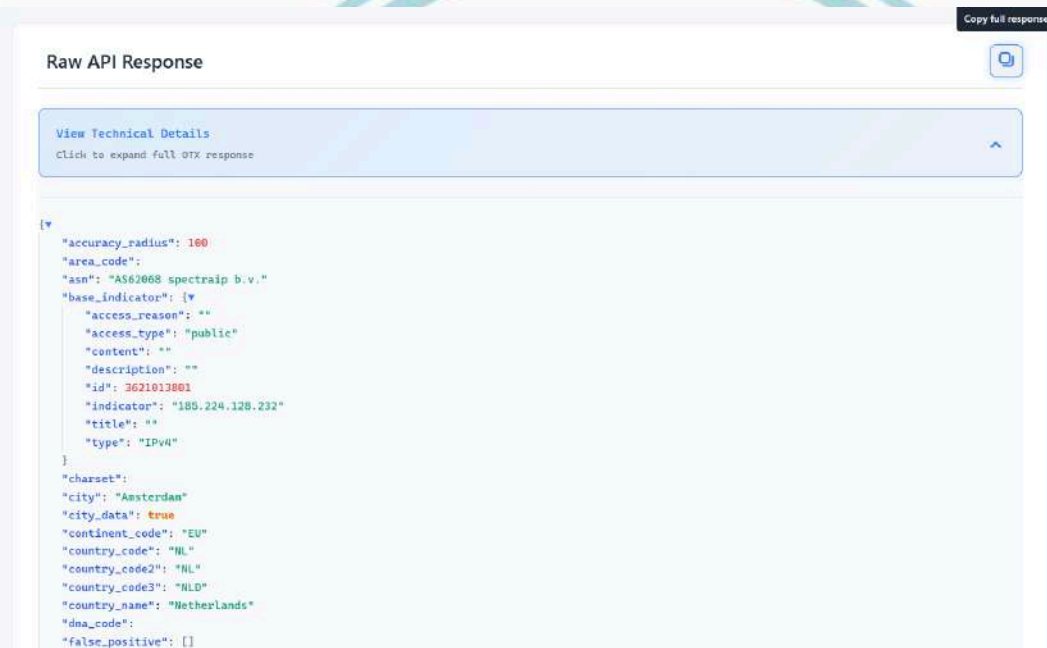


Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

OTX. Alih-alih menampilkan teks datar (*plain text*) yang sulit dibaca, komponen ini merender dokumen JSON menjadi format struktur pohon yang terwarnai.

Analisis dapat memperluas atau menutup setiap *node* data yang sifatnya *nested* ini untuk menemukan indikator spesifik dengan cepat. Lebih jauh, komponen ini dilengkapi dengan tombol *Copy to Clipboard* sekali klik, mempercepat alur kerja analisis saat harus memindahkan bukti forensik tersebut ke dalam laporan insiden (*incident response ticket*).



Gambar 4.65 Komponen Interaktif JSON Document Explorer

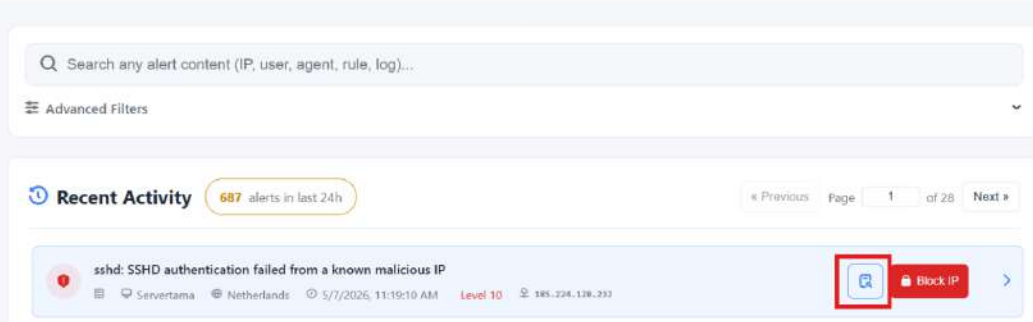
Hasil akhir dari seluruh proses asinkron di *backend* disajikan melalui antarmuka visual yang dirancang untuk meminimalkan beban kognitif analisis. Untuk memaksimalkan kecepatan investigasi, arsitektur antarmuka sistem dibangun agar saling terintegrasi erat antar-modul.

Alih-alih memaksa analisis untuk menyalin alamat IP secara manual, sistem memfasilitasi alur kerja investigasi instan (*Click-to-Investigate*) langsung dari halaman *Dashboard* utama. Seperti yang terlihat pada Gambar 4.66, sistem menginjeksi tombol pintasan berlambang kaca pembesar (*Lookup*) tepat di samping peringatan keamanan yang mengandung entitas *Indicator of Compromise* (IoC) seperti alamat IP.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 4.66 Tombol *Lookup* untuk Shortcut Intelligence pada *Dashboard*

4.3.5 Implementasi Pemantauan Proaktif dan Notifikasi *Real-Time* (Telegram)

Pendekatan pemantauan keamanan tradisional umumnya bersifat reaktif, di mana administrator harus secara aktif membuka dan melihat layar *dashboard* untuk mengetahui apakah infrastruktur Wazuh gagal terhubung atau mengalami *error*. Untuk memenuhi kebutuhan postur keamanan yang proaktif, sistem dilengkapi dengan modul pekerja latar belakang (*background worker*) yang bertindak sebagai pengawas independen, yang selanjutnya disebut sebagai *Sentinel*.

Modul ini secara kontinu memantau ketersediaan (*availability*) seluruh komponen terdaftar dan mengirimkan peringatan seketika (*real-time*) jika terjadi anomali, tanpa membebani antarmuka pengguna web.

A. Alur Sekuensial Pemantauan Latar Belakang Wazuh (*Background Scheduler*)

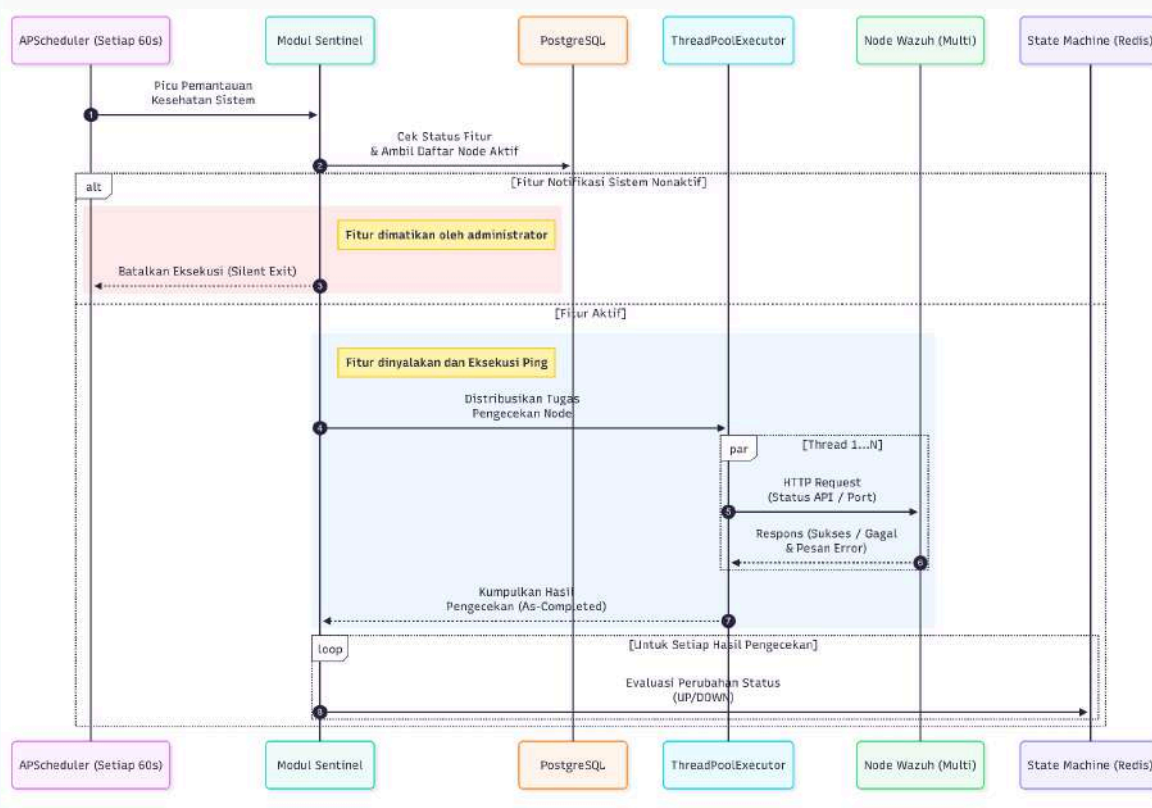
Modul pemantauan (*background scheduler*) ini menjadi dasar pembangunan sistem notifikasi *real-time* dan karenanya dibangun sebelum membuat sistem notifikasi. Modul ini diorkestrasi menggunakan *library* APScheduler yang berjalan pada *thread* terpisah dari aplikasi web utama. Rutinitas pengecekan dieksekusi secara otomatis setiap 60 detik. Untuk memastikan proses *ping* ke banyak *server* tidak memakan waktu lama, sistem menerapkan eksekusi komputasi paralel.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Alur pemantauan proaktif ini dimodelkan pada diagram sekuensial berikut:



Gambar 4.67 Diagram Alur Sekuensial Pemantauan Latar Belakang (*Sentinel*)

Penjelasan dari alur sekuensial pemantauan latar belakang untuk komponen Wazuh tersebut dapat dijelaskan sebagai fase-fase berikut:

1. Fase Awal dan Validasi Konfigurasi (Langkah 1-3): sistem memulai siklus pemantauan ketika *timer* memicu proses otomatis. Sebelum melakukan komunikasi dengan layanan eksternal, sistem terlebih dahulu memverifikasi konfigurasi fitur pada *database* sebagai mekanisme *fail-safe*. Apabila fitur notifikasi dinonaktifkan oleh administrator, proses akan dihentikan secara langsung guna menghindari penggunaan sumber daya yang tidak diperlukan.
2. Fase Distribusi dan Eksekusi Pemantauan Paralel (Langkah 4-7): Ketika fitur dalam kondisi aktif, sistem mendistribusikan tugas pengecekan konektivitas ke *Thread Pool Executor*. Setiap *thread* kemudian melakukan



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

permintaan koneksi secara paralel terhadap *node* Wazuh, baik *Manager* maupun *Indexer*, sehingga proses pemantauan dapat berlangsung lebih efisien dan simultan.

3. Fase Konsolidasi dan Evaluasi Status *Node* (Langkah 8): seluruh hasil pengecekan yang telah dikembalikan oleh masing-masing utas dikumpulkan kembali oleh modul *Sentinel* untuk dievaluasi melalui *State Machine*. Tahap ini bertujuan mengidentifikasi perubahan kondisi layanan, seperti transisi status UP atau DOWN, sebagai dasar pengambilan tindakan lanjutan oleh sistem.

Implementasi teknis pengumpulan data paralel pada modul *Sentinel* ini dapat dilihat pada potongan kode berikut:

```

1  # Potongan Kode Fungsi Utama Pemantauan
2  from concurrent.futures import ThreadPoolExecutor, as_completed
3
4  def monitor_system_health(app):
5      with app.app_context():
6          # 1. Fail-Safe: Batalkan eksekusi jika fitur notifikasi
          dimatikan oleh Admin
7          settings = TelegramSettings.query.first()
8          if not settings or not
          settings.system_notifications_enabled:
9              return
10
11         # 2. Mengambil seluruh komponen infrastruktur yang berstatus
          aktif
12         active_indexers =
          WazuhIndexer.query.filter_by(is_active=True).all()
13         active_managers =
          WazuhManager.query.filter_by(is_active=True).all()
14         if not active_indexers and not active_managers: return
15
16         # 3. Menyiapkan eksekusi paralel menggunakan
          ThreadPoolExecutor
17         total_workers = len(active_indexers) + len(active_managers)
18         futures = []
19
20         with ThreadPoolExecutor(max_workers=total_workers) as
          executor:
21             # 4. Mendistribusikan tugas ping Indexer dan Manager ke
              dalam thread terpisah
22             for idx in active_indexers:
23                 futures.append(executor.submit(_check_indexer_worker, app, idx))

```

**Hak Cipta :**

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

24         for mgr in active_managers:
25             futures.append(executor.submit(_check_manager_worker, app, mgr))
26
27             # 5. Menangkap hasil dari setiap thread yang telah
                selesai bertugas
28             for future in as_completed(futures):
29                 try:
30                     comp_type, comp_id, comp_name, is_ok, error_msg
31                     = future.result()
32
33                     # 6. Menyerahkan hasil ping ke State Machine
                        untuk dievaluasi
34                     _evaluate_component_state(
35                         app.redis, comp_type, comp_id, comp_name,
36                         is_ok, error_msg
37                     )
38                     except Exception as e:
39                         app.logger.error(f"Error memproses thread
                        Sentinel: {e}")

```

Gambar 4.68 Potongan Kode Fungsi Utama Pemantauan

B. Implementasi Logika *State Machine* dan Konektor API Telegram Bot

Keberhasilan modul pemantauan latar belakang (*Sentinel*) dalam mendeteksi status komponen Wazuh *Indexer* dan Wazuh *Manager* secara berkelanjutan (sebagaimana diuraikan pada Bagian A) menjadi fondasi utama bagi sistem peringatan dini. Setelah adanya mekanisme deteksi, implementasi krusial selanjutnya adalah menentukan kapan sistem harus bertindak.

Tantangan terbesar dalam sistem pemantauan yang berjalan setiap 60 detik adalah risiko *alert fatigue* (kelelahan akibat banyaknya notifikasi). Jika sebuah *server* Wazuh terputus selama satu jam, administrator tidak boleh menerima 60 pesan peringatan yang sama secara beruntun tiap menitnya.

Untuk mencegah hal ini, sebelum merancang konektor pengiriman pesan, sistem terlebih dahulu mendefinisikan logika keadaan mesin (*state machine*) di dalam memori volatil Redis. Logika ini bertindak sebagai "otak" *filter* yang mengevaluasi hasil *polling* dari *Sentinel* ke dalam empat skenario keputusan mutlak. Tabel 4.17 berikut adalah implementasi matriks keputusan pada fungsi evaluasi *Sentinel*:



Tabel 4.17. Tabel Daftar Skenario dan Keadaan *State Machine*

No	Skenario	Penjelasan Keadaan
1	Skenario A (<i>Baseline</i>)	Jika komponen masih terhubung (UP ke UP), sistem diam (tidak melakukan aksi basis data maupun notifikasi).
2	Skenario B (Insiden)	Jika koneksi tiba-tiba terputus (UP ke DOWN), sistem mencatat waktu kejadian (<i>first_down_at</i>) dan memicu notifikasi kritis seketika.
3	Skenario C (<i>Ongoing Outage</i>)	Jika komponen belum pulih (DOWN ke DOWN), sistem dengan cerdas menahan pengiriman pesan. Sistem hanya akan mempublikasikan pesan pengingat (<i>Reminder</i>) jika durasi pemadaman telah mencapai kelipatan 3 jam. Hal ini menjaga kanal komunikasi administrator tetap bersih dari <i>spam</i> .
4	Skenario D (Resolusi)	Jika komponen kembali <i>online</i> (DOWN ke UP), sistem mempublikasikan pesan pemulihan (<i>Recovered</i>) dan mereset status <i>State Machine</i> kembali ke <i>normal</i> .

Berdasarkan matriks keputusan di atas, sistem merekam *timestamp* status terakhir setiap komponen. Apabila status memenuhi syarat transisi pada Skenario B, C, atau D, barulah sistem memicu event peringatan. Implementasi teknis dari logika *State Machine* ini ditunjukkan pada potongan kode Gambar 4.69.

```

1 # Potongan Kode Fungsi Evaluasi Status UP DOWN Komponen
2 def _evaluate_component_state(redis_client, comp_type, comp_id,
3   comp_name, is_ok, error_msg):
4     redis_key = f"sys:sentinel:{comp_type.lower()}:{comp_id}"
5     # 1. Mengambil status komponen terakhir dari memori Redis
6     (Default: UP)
7     state = json.loads(redis_client.get(redis_key)) if
8     redis_client.get(redis_key) else {"status": "UP"}

```



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

7     previous_status = state["status"]
8
9     # SKENARIO A: Sistem Normal (UP ke UP) -> Perbarui masa aktif
    (TTL) key Redis saja
10    if is_ok and previous_status == "UP":
11        redis_client.expire(redis_key, 604800) # 7 Hari TTL untuk
    mencegah memory leak
12        return
13
14    # SKENARIO B: Insiden Dimulai (UP ke DOWN) -> Kirim Alert
    CRITICAL
15    elif not is_ok and previous_status == "UP":
16        state.update({"status": "DOWN", "first_down_at": now,
    "last_error": error_msg})
17        _publish_event("sentinel_down", comp_name, error_msg) #
    Lempar ke Telegram
18
19    # SKENARIO C: Pemadaman Berlangsung (DOWN ke DOWN) -> Tahan
    notifikasi (Cegah Spam)
20    elif not is_ok and previous_status == "DOWN":
21        state["last_error"] = error_msg # Selalu perbarui alasan
    error terbaru
22
23        # Logika Peningat (Reminder): Kirim notifikasi ulang HANYA
    jika sudah lewat 3 jam
24
25        elapsed_seconds = (now_utc - last_alert_dt).total_seconds()
26        if elapsed_seconds >= 10800: # 10800 detik = 3 jam
27            state["last_alert_at"] = now_utc
28            _publish_event("sentinel_realert", comp_name, error_msg)
29
30    # SKENARIO D: Pemulihan Jaringan (DOWN ke UP) -> Kirim Alert
    RECOVERED
31    elif is_ok and previous_status == "DOWN":
32        _publish_event("sentinel_recovered", comp_name, "")
33        state.update({"status": "UP", "first_down_at": None,
    "last_error": None})
34
35    # 2. Menyimpan state terbaru kembali ke Redis
36    redis_client.setex(redis_key, 604800, json.dumps(state))
  
```

Gambar 4.69 Potongan Kode Fungsi Evaluasi Status UP DOWN Komponen

Selain itu, setiap kunci (*key*) status di dalam Redis diberikan batas waktu hidup (TTL) selama 7 hari (604800 detik). Ini berfungsi sebagai mekanisme *garbage collection*. Jika Administrator menghapus suatu komponen dari sistem



Hak Cipta :

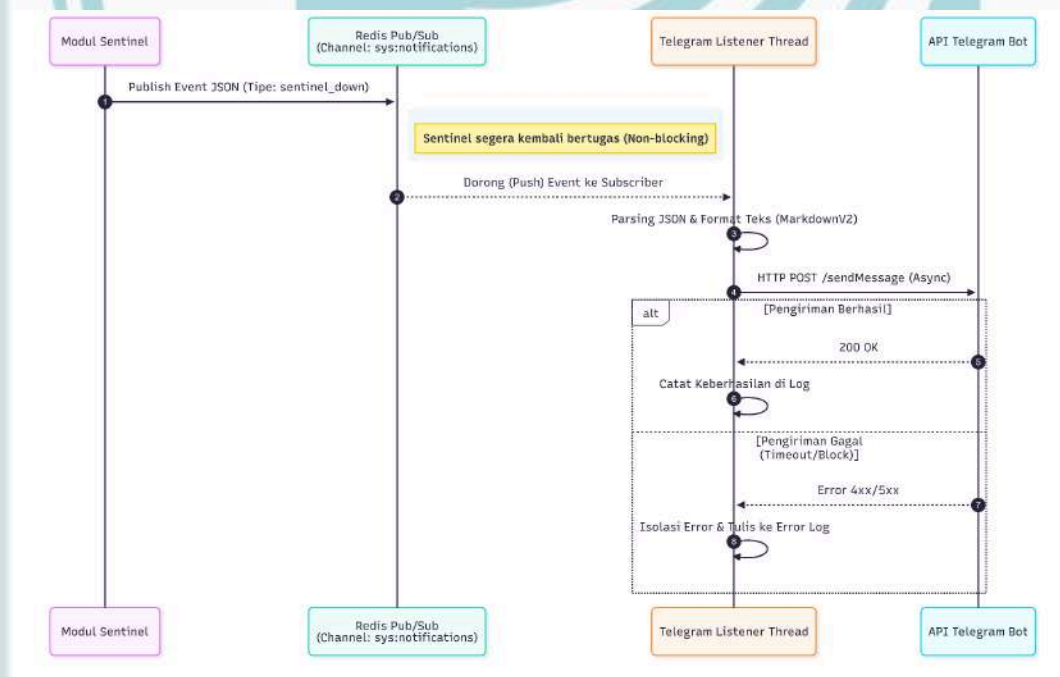
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

secara permanen, kunci status komponen tersebut di memori tidak akan menjadi *ghost state*, melainkan akan menghilang dengan sendirinya setelah 7 hari.

C. Mekanisme Komunikasi Telegram Asinkron (*Publish/Subscribe*)

Ketika *State Machine* (Skenario B, C, atau D pada tahap sebelumnya) memutuskan bahwa sebuah peristiwa layak untuk diberitahukan, modul *Sentinel* tidak serta-merta mengeksekusi pemanggilan langsung ke API Telegram. Mengirim request jaringan ke API eksternal sangat rentan terhadap latensi dan *timeout*, yang dapat menyebabkan pekerja latar belakang (*scheduler*) terhenti (*hang*).

Untuk mencegah hal tersebut, sistem mengadopsi pola arsitektur terpisah (*Decoupled Architecture*) dengan memanfaatkan fitur *Publish/Subscribe* (Pub/Sub) pada Redis. Alur pengiriman pesan ini dimodelkan pada diagram sekuensial berikut:



Gambar 4.70 Diagram Alur Sekuensial Pengiriman Pesan Asinkron (Pub/Sub)

Berdasarkan diagram di atas, *Sentinel* hanya bertugas mengonstruksi sebuah *payload* JSON yang berisi informasi kejadian (seperti nama komponen dan pesan *error*), lalu mempublikasikannya (*publish*) ke saluran khusus di Redis



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

(Langkah 1). Setelah itu, *Sentinel* dapat segera melanjutkan tugas pemantauannya tanpa harus menunggu balasan dari Telegram. Di sisi lain, sebuah *Listener Thread* yang beroperasi menerus akan menangkap pesan tersebut (Langkah 2), merakit struktur pesannya, dan memanggil Konektor API Telegram.

Konektor ini diimplementasikan menggunakan *library requests* di Python untuk mengeksekusi HTTP *POST* langsung ke *server* Telegram dengan detail implelementasi seperti Gambar 4.71.

```

1 # Potongan Kode Konektor API Telegram
2 def _send_telegram_message(message: str, bot_token: str,
  chat_ids_string: str, logger) -> bool:
3     # 1. Mem-parsing string input menjadi list Chat ID dan
    membersihkan spasi
4     chat_id_list = [cid.strip() for cid in
  chat_ids_string.split(",") if cid.strip()]
5     if not chat_id_list:
6         return False
7
8     # 2. Mengkonstruksi URL API Telegram secara dinamis menggunakan
    Bot Token
9     api_url = f"https://api.telegram.org/bot{bot_token}/sendMessage"
10    success_count = 0
11
12    # 3. Iterasi pengiriman pesan ke setiap Chat ID secara
    sekuensial
13    for chat_id in chat_id_list:
14        payload = {
15            "chat_id": chat_id,
16            "text": message,
17            . . .
18        }
19
20        try:
21            # 4. Mengeksekusi HTTP POST dengan batas waktu 10 detik
22            response = requests.post(api_url, json=payload,
  timeout=10)
23            response.raise_for_status()
24            success_count += 1
25
26        except requests.exceptions.RequestException as e:
27            # 5. Toleransi kesalahan: Abaikan error pada satu Chat
    ID agar iterasi berlanjut
28            logger.error(f"Gagal mengirim pesan ke {chat id}: {e}")

```

Gambar 4.71 Potongan Kode Konektor API Telegram



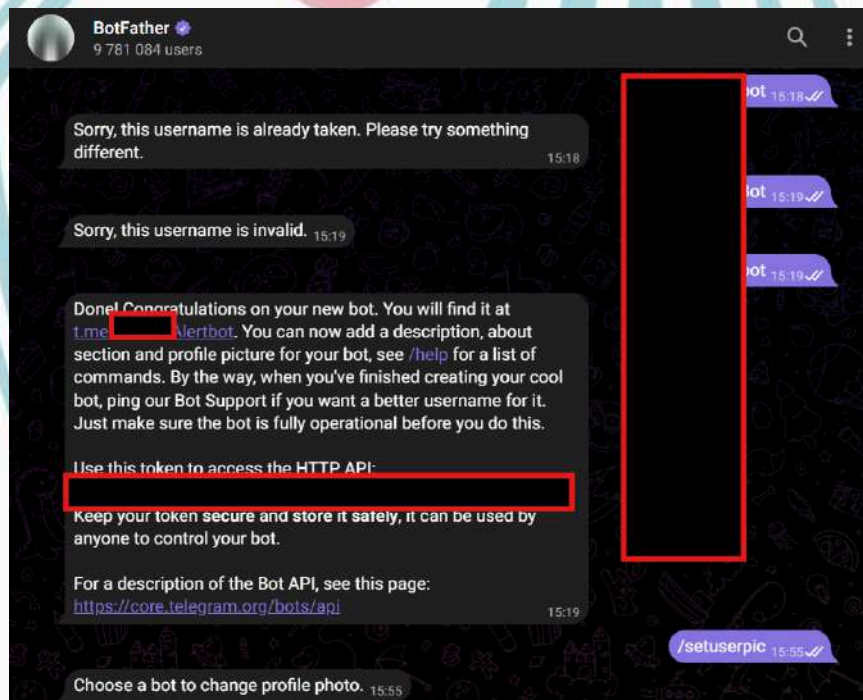
Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Pendekatan iterasi mandiri di dalam blok *try-except* seperti pada kode menjamin *fault tolerance*. Apabila pengiriman pesan gagal pada satu *Chat ID* (misalnya karena pengguna memblokir bot), proses eksekusi akan terus berjalan untuk *Chat ID* berikutnya.

D. Manajemen Konfigurasi dan Pengujian Rute Pesan

Sebelum konektor API di atas dapat bekerja dan pesan dapat dikirimkan ke perangkat analis keamanan, sistem membutuhkan sebuah agen perantara. Organisasi atau Administrator harus terlebih dahulu membuat sebuah bot Telegram. Proses pembuatan bot ini difasilitasi langsung oleh platform Telegram melalui antarmuka resmi *@BotFather* seperti pada gambar 4.72.



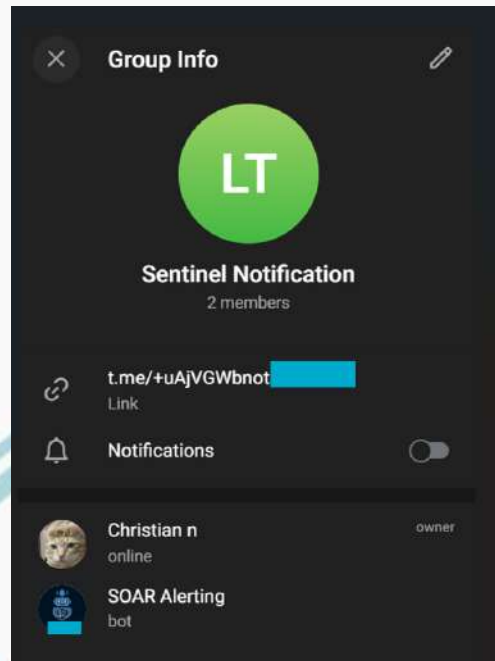
Gambar 4.72 Tampilan Pembuatan Bot Telegram dengan BotFather

Apabila proses registrasi bot telah selesai dan *token ID* yang bertindak sebagai kunci autentikasi sudah berhasil didapatkan, tahapan selanjutnya berfokus pada persiapan tempat menerima pesan. Pengguna dapat membentuk sebuah grup Telegram khusus yang didedikasikan sebagai saluran utama untuk menerima siaran informasi. *Chat ID* dapat didapatkan dari *link* url saat grup dibuka pada browser.



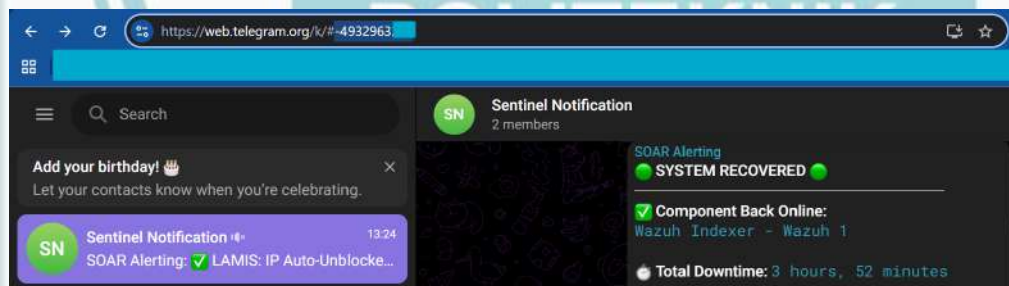
Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 4.73 Tampilan Grup Telegram

Untuk memperoleh *Chat ID*, Administrator dapat menggunakan metode yang paling praktis, yaitu dengan membuka grup melalui Telegram Web dan mengambil nilai *Chat ID* yang terdapat pada URI halaman. Contoh tampilannya ditunjukkan pada Gambar 4.74.



Gambar 4.74 Tampilan URL untuk Mendapatkan *Chat ID* Grup Telegram

Setelah proses tersebut selesai, Administrator akan memperoleh sebuah *Bot Token* rahasia yang berfungsi sebagai kredensial autentikasi untuk mengakses HTTP API Telegram. Token ini menjadi identitas utama yang digunakan sistem untuk mengirimkan permintaan ke layanan Telegram.

Selanjutnya, *Bot Token* tersebut harus didaftarkan ke dalam sistem manajemen bersama daftar *Chat ID*, yaitu identitas grup atau pengguna yang akan menjadi tujuan pengiriman notifikasi. Kedua informasi tersebut diisikan pada *field*



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

konfigurasi yang tersedia di dalam sistem, sebagaimana ditunjukkan pada Gambar 4.75 berikut.

The screenshot shows a 'Configuration' modal with the following fields and controls:

- Bot Token:** A text input field with a placeholder 'Enter new token to update'. A small note says 'Leave blank to keep current'.
- Chat IDs:** A text input field containing '-123456789'. A note below says 'Comma-separated list of chat IDs. Add any Telegram user info bot to your group or chat to get the chat ID.'
- SAVE SETTINGS:** A blue button.
- Send Test Notification:** A checkbox that is checked.

Gambar 4.75 Tampilan *Field Configuration* untuk Input Bot *Token* dan *Chat IDs*

Serupa dengan keamanan kredensial pada modul lainnya, Bot *Token* yang diinputkan pada Gambar 4.76 akan langsung dienkripsi menggunakan algoritma AES (Security by Design) sebelum disimpan ke dalam basis data PostgreSQL. Saat penyimpanan berhasil maka akan ada *toast notification* berwarna hijau yang menandakan sukses.

The screenshot shows the 'Telegram Integration' page with a success toast and configuration options:

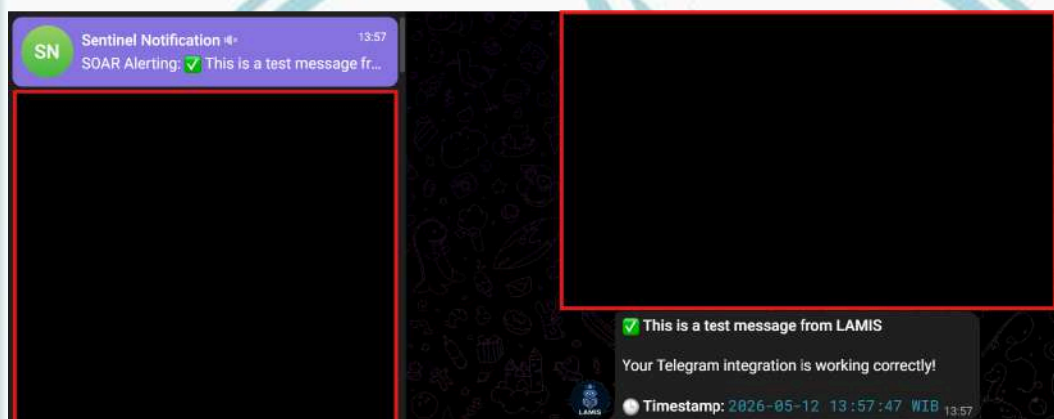
- Telegram Integration:** Configure Telegram notifications for real-time alerts on IP blocks and unblocks.
- Telegram settings saved successfully:** A green toast notification.
- Information:**
 - Telegram Integration is Active:** A green status bar with a checkmark icon. Text: 'Your Bot Token and Chat IDs are configured. Notifications will be sent based on your settings below.'
- Notification Settings:**
 - Attack Notifications:** Toggle switch (ON). Text: 'Send notifications for high severity Watch alerts.'
 - IP Block Notifications:** Toggle switch (ON). Text: 'Send notifications when IPs are blocked or unblocked.'
 - System Notifications:** Toggle switch (ON). Text: 'Send notifications for LAMIS Sentinel health events.'
- Configuration:**
 - Bot Token:** A text input field with a placeholder 'Enter new token to update'. A small note says 'Leave blank to keep current'.
 - Chat IDs:** A text input field containing '-493296'. A note below says 'Comma-separated list of chat IDs. Add any Telegram user info bot to your group or chat to get the chat ID.'
 - SAVE SETTINGS:** A blue button.
 - Send Test Notification:** A checkbox that is checked.

Gambar 4.76 Tampilan *Toast Sukses* dan Keseluruhan Page Telegram Integration

**Hak Cipta :**

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Untuk mencegah insiden kegagalan notifikasi akibat kesalahan ketik saat memasukkan *Token* atau *Chat ID*, sistem menyediakan fitur pengujian rute (*Send Test Notification*). Ketika tombol pengujian ditekan, antarmuka web memicu request AJAX secara asinkron ke *backend*. Sistem akan mengirimkan pesan *dummy* ke API Telegram dan langsung merender hasilnya di layar menggunakan elemen *Toast Notification*. Jika konfigurasi tidak valid, pesan *error* teknis akan ditampilkan dengan rapi, memastikan jalur komunikasi benar-benar valid sebelum sistem beroperasi penuh.



Gambar 4.77 Tampilan Notifikasi Test Message

E. Umpan Balik Visual dan Peringatan Kegagalan Kritis

Keberhasilan modul Sentinel yang berjalan di latar belakang tidak akan memberikan manfaat optimal tanpa mekanisme penyampaian informasi yang cepat dan mudah dipahami oleh analis keamanan. Oleh karena itu, sistem merancang mekanisme umpan balik peringatan pada dua platform yang saling melengkapi untuk memastikan setiap insiden dapat segera diketahui. Pendekatan ini menyediakan redundansi mekanisme notifikasi sehingga informasi tetap dapat diterima meskipun salah satu media mengalami kendala.

Sebelum peringatan dikirim ke aplikasi Telegram milik analis, *Listener Thread* terlebih dahulu mengambil *payload* JSON dari Redis, kemudian mengekstrak informasi yang diperlukan dan menyusunnya menjadi pesan teks yang terstruktur. Selama proses ini, sistem menerapkan format MarkdownV2 Telegram agar elemen-elemen penting pada notifikasi memiliki hierarki visual



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

yang jelas dan lebih mudah dibaca oleh penerima. Potongan kode Gambar 4.78 menunjukkan proses pembentukan format pesan tersebut.

```

1 # Potongan Kode Fungsi Pemformatan Teks Telegram
2 def send_system_health_notification(payload, event_type):
3     # 1. Mengekstrak dan membersihkan (escape) data dari payload
4     comp_type = _escape_markdown(payload.get("component_type"))
5     comp_name = _escape_markdown(payload.get("component_name"))
6     error_msg = _escape_markdown(payload.get("error_message"))
7     timestamp_wib = _escape_markdown(_get_wib_timestamp())
8
9     # 2. Logika Kalkulasi Downtime (Jika ada data waktu
    putus/first_down_at)
10    duration_str = "Unknown"
11    if payload.get("first_down_at"):
12        first_down_at =
    datetime.fromisoformat(payload["first_down_at"]).replace(tzinfo=timezone
    .utc)
13        duration = datetime.now(timezone.utc) - first_down_at
14        duration_str = _escape_markdown(_format_duration(duration))
15
16    # 3. Merakit Teks berdasarkan Tipe Event (DOWN vs RECOVERED)
17    if event_type == "sentinel_down":
18        message = (
19            f"🚨 *CRITICAL HEALTH ALERT* 🚨\n"
20            f"🔥 *Component Offline:* \n`{comp_type}` \n-
    {comp_name}` \n\n"
21            f"⚠️ *Failure Details:* \n`{error_msg}` \n\n"
22            f"🕒 *Detected At:* `{timestamp_wib}`"
23        )
24    elif event_type == "sentinel_recovered":
25        message = (
26            f"✅ *SYSTEM RECOVERED* ✅\n"
27            f"🟢 *Component Back Online:* \n`{comp_type}` \n-
    {comp_name}` \n\n"
28            f"🕒 *Total Downtime:* `{duration_str}` \n\n"
29            f"🕒 *Recovered At:* `{timestamp_wib}`"
30        )
31
32    # 4. Mendelegasikan pengiriman teks final ke konektor API
    Telegram
33    _send_telegram_message(message)
  
```

Gambar 4.78 Potongan Kode Fungsi Pemformatan Teks Telegram

Pada potongan kode Gambar 4.78 bagian #3, sistem sengaja menggunakan pictogram visual (seperti merah dan hijau), teks tebal (*bold*), dan format blok kode (*monospace/code block* untuk nama komponen dan *error*) agar analis dapat



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

langsung mengidentifikasi tingkat keparahan insiden dalam sepersekian detik di layar ponsel mereka.

Untuk memvalidasi keandalan dan akurasi logika *Sentinel* serta pemformatan pesan di atas, sebuah skenario pengujian (*experiment*) dilakukan. Pengujian disimulasikan dengan memutus koneksi *server* Wazuh target secara paksa menggunakan antarmuka baris perintah (CLI) Docker, menyimulasikan pemadaman *server* secara tiba-tiba (*sudden outage*) seperti pada gambar berikut.

```
chris@ [REDACTED] ~/wazuh-docker/single-node$ docker compose pause
WARN[0000] /home/chris/wazuh-docker/single-node/docker-compose.yml: the attribute `version` is obsolete, it
will be ignored, please remove it to avoid potential confusion
[+] Pausing 2/2
✓ Container single-node-wazuh.indexer-1 Paused 0.0s
✓ Container single-node-wazuh.manager-1 Paused 0.0s
chris@ [REDACTED] ~/wazuh-docker/single-node$ docker compose unpause
WARN[0000] /home/chris/wazuh-docker/single-node/docker-compose.yml: the attribute `version` is obsolete, it
will be ignored, please remove it to avoid potential confusion
[+] Running 2/2
✓ Container single-node-wazuh.indexer-1 Unpaused 0.0s
✓ Container single-node-wazuh.manager-1 Unpaused 0.0s
```

Gambar 4.79 Perintah CLI untuk Mensimulasikan Skenario B (Pemadaman *Server Target*)

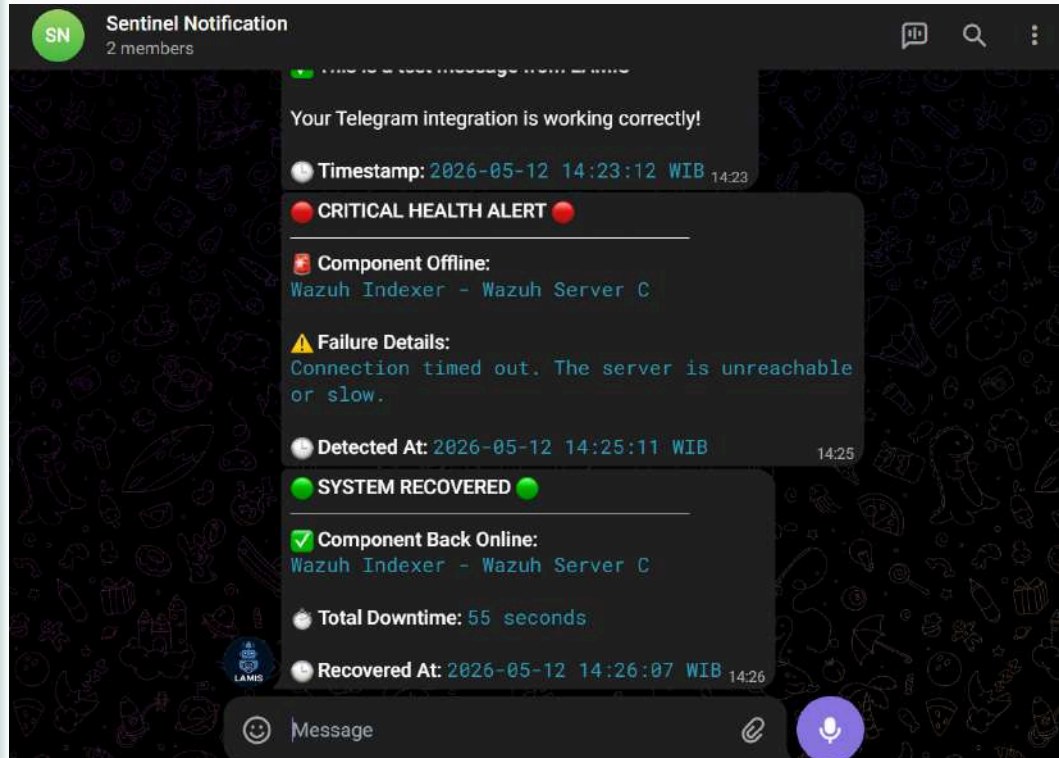
Dalam waktu kurang dari 60 detik setelah perintah pada Gambar 4.79 dieksekusi, modul *Sentinel* tidak lagi menerima balasan ping dari *server* target. Kondisi tersebut diinterpretasikan sebagai indikasi bahwa layanan tidak lagi tersedia, sehingga sistem secara otomatis berpindah ke Skenario B (Insiden Dimulai) sesuai dengan *State Machine* yang telah dirancang. Tanpa memerlukan intervensi administrator, sistem segera mengirimkan pesan peringatan ke grup Telegram analis agar informasi mengenai gangguan dapat diterima sesegera mungkin.

Setelah kondisi pemadaman berhasil dikonfirmasi, perintah pemulihan (*docker compose unpause*) dijalankan melalui CLI untuk mengaktifkan kembali layanan Wazuh.. Modul *Sentinel* mendeteksi kembalinya respons HTTP 200 OK dan segera memicu Skenario D (Resolusi). Hasil dari kedua skenario tersebut divisualisasikan secara langsung pada aplikasi Telegram sehingga analis dapat memantau perubahan status layanan secara real time tanpa perlu mengakses dashboard sistem seperti pada Gambar 4.80.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 4.80 Umpan Balik Peringatan Kritis dan Pemulihan pada Telegram

Seperti yang ditunjukkan pada Gambar 4.81, sistem memberikan konteks insiden yang sangat komprehensif. Pada pesan ‘CRITICAL HEALTH ALERT’, analis dapat melihat secara spesifik komponen mana yang mati (*Wazuh Indexer - Wazuh Server C*) dan alasan teknisnya (*Connection timed out. The server is unreachable or slow*).

Lebih jauh lagi, pada pesan ‘SYSTEM RECOVERED’, sistem secara cerdas menghitung dan melaporkan durasi pemadaman (*Total Downtime: 55 seconds*). Data ini sangat krusial bagi manajer SOC dalam mengukur Service Level Agreement (SLA) keandalan infrastruktur keamanan mereka.

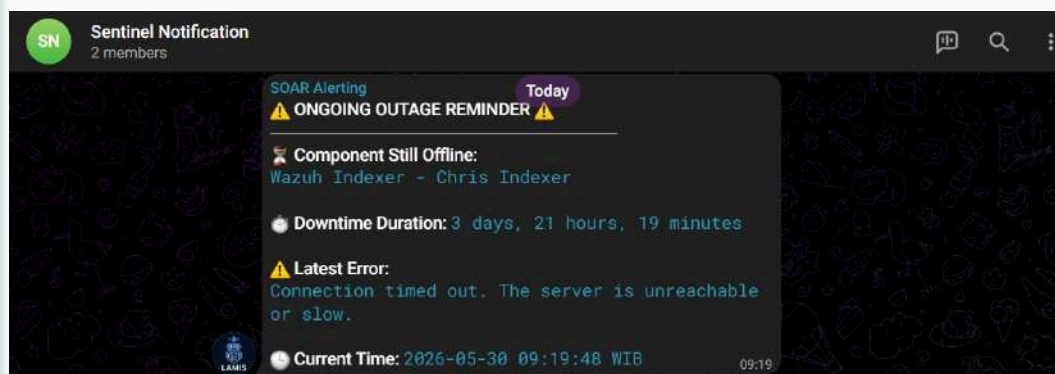
Dalam rentang waktu antara ‘CRITICAL HEALTH ALERT’ hingga ‘SYSTEM RECOVERED’, terdapat mekanisme cerdas yang memonitor durasi pemadaman sistem secara berkesinambungan tanpa memicu *spam* notifikasi (*Alert Fatigue*). Mekanisme ini diwujudkan melalui Skenario *Ongoing Outage* pada *State Machine* di dalam memori Redis. Hanya ketika pemadaman telah melewati



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

kelipatan interval waktu tertentu, dalam sistem ini dikonfigurasi selama 3 jam (10.800 detik), sistem akan mempublikasikan pesan pengingat.



Gambar 4.81 Peringatan Berkala (*Reminder*) pada Insiden Pemadaman Jangka Panjang

Seperti yang ditunjukkan pada Gambar 4.82, sistem mengeksekusi fitur *Ongoing Outage Reminder*. Ketika *Sentinel* mendeteksi bahwa komponen masih dalam keadaan mati secara berturut-turut (DOWN ke DOWN), *server* tidak akan mengirimkan pesan pada setiap siklus *polling* 60 detik. Sebaliknya, sistem menghitung selisih waktu dari peringatan terakhir (*last_alert_at*).

Selain menyediakan notifikasi jarak jauh, sistem juga mengantisipasi kemungkinan modul *Sentinel* mengalami *crash* atau berhenti beroperasi. Untuk mencegah kondisi *silent failure*, *backend* mengirimkan sinyal *heartbeat* ke Redis setiap 60 detik. Selama sinyal tersebut tersedia, aplikasi web menganggap *Sentinel* berfungsi normal. Sebaliknya, jika *heartbeat* tidak lagi terdeteksi karena telah kedaluwarsa, aplikasi secara otomatis menampilkan *banner* peringatan berwarna kuning pada seluruh halaman dashboard sebagai indikasi bahwa modul *Sentinel* memerlukan perhatian.



Gambar 4.82 Banner Peringatan Kegagalan Proses Pemantauan Latar Belakang



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Melalui banner peringatan pada Gambar 4.82, analis akan segera disadarkan bahwa data yang mereka lihat di layar *dashboard* mungkin sudah usang, dan fitur notifikasi proaktif sedang tidak beroperasi. Integrasi peringatan ganda ini menciptakan arsitektur pemantauan yang resilien, memastikan bahwa tidak ada kegagalan infrastruktur yang luput dari perhatian organisasi.

4.3.6 Implementasi Hak Akses Berbasis Peran (*Role-Based Access Control*)

Seluruh modul yang telah diimplementasikan, mulai dari manajemen infrastruktur multi-Wazuh, pengayaan *Threat Intelligence*, hingga notifikasi status sistem, diintegrasikan ke dalam satu antarmuka terpadu. Mengingat adanya berbagai tingkat peran di lingkungan *Security Operations Center* (SOC), sistem menerapkan mekanisme *Role-Based Access Control* (RBAC). Sebagai penerapan tambahan di luar modul utama penelitian, RBAC tetap berperan penting dalam menerapkan prinsip *least privilege*, yaitu memastikan setiap pengguna hanya memperoleh hak akses sesuai dengan tugas dan tanggung jawabnya.

Sebelum hak akses diterapkan, setiap pengguna terlebih dahulu melakukan autentikasi melalui halaman *login* menggunakan kombinasi *username* dan *password* seperti pada Gambar 4.83. Setelah kredensial berhasil diverifikasi, sistem akan mengidentifikasi peran yang dimiliki pengguna dan membangun *session* sesuai dengan hak akses tersebut. Informasi peran selanjutnya digunakan untuk menentukan fitur dan menu yang dapat diakses oleh setiap pengguna.

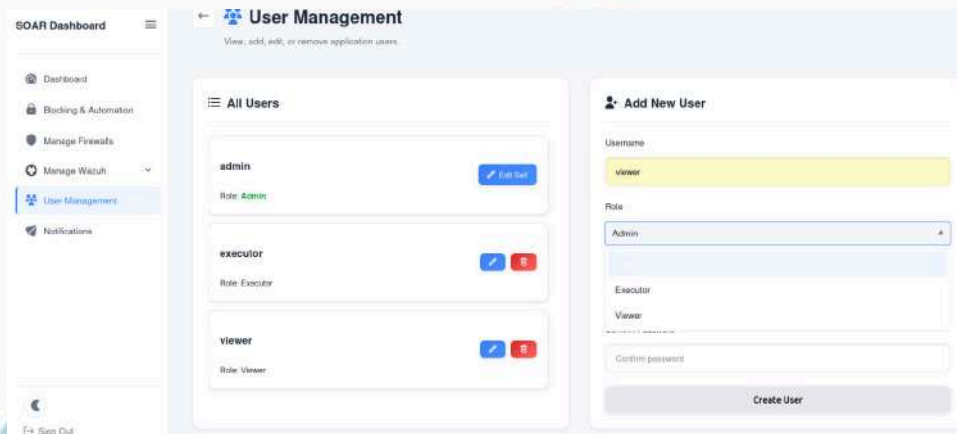
Gambar 4.83 Halaman Login Sistem Hub Manajemen Terpusat



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

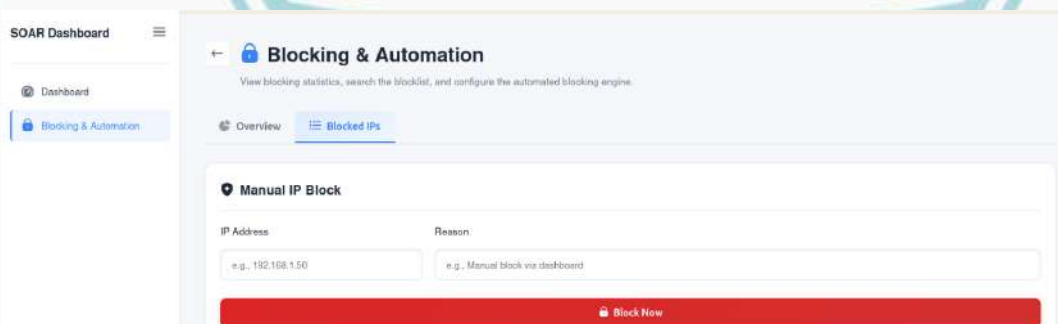
Sistem membagi hak akses ke dalam tiga peran, yaitu Administrator, *Executor*, dan *Viewer*. Masing-masing peran memiliki ruang lingkup kewenangan yang berbeda. Panel konfigurasi untuk melihat daftar pengguna yang terdaftar beserta rolenya seperti pada Gambar 4.84.



Gambar 4.84 Tampilan Antarmuka Manajemen Pengguna Khusus Administrator

Peran Administrator memiliki hak akses penuh terhadap konfigurasi sistem dengan keseluruhan menu panel terbuka seperti Gambar 4.84. Administrator bertanggung jawab mengelola seluruh kredensial Wazuh, AlienVault OTX, bot Telegram, serta mengelola akun pengguna beserta pembagian perannya.

Berbeda dengan Administrator, peran *Executor* difokuskan pada aktivitas operasional sehari-hari. Pengguna dengan peran ini dapat mengakses dua menu panel yakni *Dashboard* dan *Blocking & Automation* seperti pada Gambar 4.85. *Executor* tidak memiliki izin untuk mengubah konfigurasi sistem maupun mengelola kredensial.



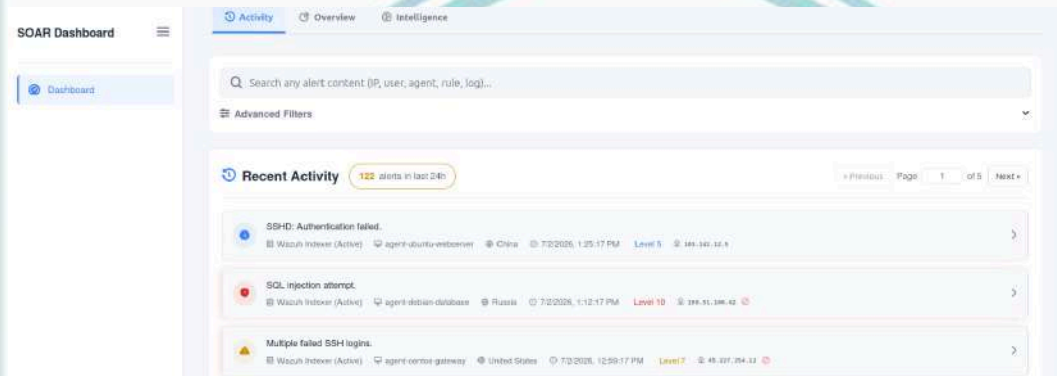
Gambar 4.85 Adaptasi Dinamis Menu Navigasi Peran *Executor*



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Sementara itu, peran *Viewer* ditujukan bagi pengguna yang hanya memerlukan akses untuk melihat informasi operasional, seperti pihak manajemen atau pemangku kepentingan. Pengguna dengan peran ini tetap dapat mengakses dashboard, status infrastruktur, dan tampilan *alerts*, tetapi tidak diberikan hak untuk melakukan pemblokiran atau perubahan konfigurasi apapun seperti terlihat pada Gambar 4.86.



Gambar 4.86 Adaptasi Dinamis Menu Navigasi Peran *Viewer*

4.4 Pengujian

Subbab ini menjelaskan proses pengujian yang dilakukan untuk memastikan sistem berjalan sesuai dengan kebutuhan dan rancangan yang telah ditetapkan. Pengujian difokuskan pada kinerja, ketepatan keluaran, dan keandalan sistem dalam berbagai kondisi. Bagian selanjutnya memaparkan deskripsi pengujian, prosedur pelaksanaan, data hasil, serta analisis dan evaluasinya.

4.4.1 Deskripsi Pengujian

Pengujian sistem pada penelitian ini dibagi menjadi tiga fase utama untuk memvalidasi kualitas perangkat lunak secara menyeluruh, mulai dari kelayakan teknis hingga dampaknya terhadap produktivitas kerja melalui tiga fase pengujian: fungsional, non-fungsional, dan komparatif efisiensi.

- Fase pertama yakni pengujian fungsional, menggunakan metode *black box* untuk memverifikasi operasional fitur pada hub manajemen terpusat. Evaluasi difokuskan pada interaksi antarmuka dengan indikator keberhasilan biner (berhasil/gagal) berdasarkan kesesuaian output sistem terhadap spesifikasi yang telah ditetapkan.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

- Fase kedua mengevaluasi aspek non-fungsional yang mencakup tiga parameter operasional utama: kinerja, keamanan, dan keandalan. Pengujian kinerja mengukur metrik waktu muat dasbor dan latensi respons sistem. Pengujian keamanan memvalidasi efektivitas mekanisme enkripsi dalam melindungi data kredensial dari akses tidak sah. Sementara itu, pengujian keandalan mengukur *fault tolerance* sistem dalam mempertahankan ketersediaan layanan saat menghadapi anomali jaringan atau kegagalan koneksi dengan layanan pihak ketiga.
- Fase ketiga, pengujian komparatif efisiensi, mengukur tingkat optimalisasi proses pemantauan keamanan secara kuantitatif. Pengujian eksperimental ini membandingkan mekanisme alur kerja manual (*baseline*) dengan sistem usulan. Peningkatan efisiensi diukur melalui dua variabel: reduksi durasi penyelesaian tugas (*time on task*) dalam satuan detik dan penurunan beban interaksi pengguna (*click/step count*). Data dianalisis menggunakan metode komparatif deskriptif.

Untuk mengkuantifikasi persentase hasil pengujian fase ketiga, penelitian ini mengadaptasi metrik *Deployment Time Reduction Rate* (DTRR) dari Hyun et al. (2024). Formulasi tersebut disesuaikan untuk membandingkan durasi operasional antara metode manual dan sistem terintegrasi, sebagaimana dijabarkan pada persamaan berikut:

$$Efisiensi = \frac{WaktuManual - WaktuTerintegrasi}{WaktuManual} \times 100\%$$

Dengan keterangan:

- Efisiensi (%) = Persentase peningkatan efisiensi waktu.
- WaktuManual = Waktu yang dibutuhkan untuk menyelesaikan suatu aktivitas menggunakan metode manual (detik).
- WaktuTerintegrasi = Waktu yang dibutuhkan untuk menyelesaikan aktivitas yang sama menggunakan sistem terintegrasi (detik).

4.4.2 Prosedur Pengujian

Untuk mendapatkan data yang akurat dan komprehensif, prosedur pengujian pada penelitian ini dijabarkan ke dalam tiga kategori utama yang saling



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

berkesinambungan. Ketiga kategori tersebut meliputi: (A) Pengujian Fungsional (*Black Box Testing*) untuk memvalidasi kesesuaian input dan output fitur; (B) Pengujian Non-Fungsional untuk mengevaluasi atribut kualitas sistem seperti kinerja, keamanan, dan keandalan; serta (C) Pengujian Efisiensi Operasional untuk mengukur tingkat optimalisasi alur kerja. Prosedur untuk masing-masing pengujian diuraikan sebagai berikut:

A. Prosedur Pengujian Fungsional (*Black Box Testing*)

Pengujian fungsional dilakukan menggunakan metode *Black Box Testing* yang dipecah menjadi berbagai skenario pengujian spesifik. Metode ini berfokus pada evaluasi antarmuka dan fungsionalitas sistem tanpa perlu meninjau struktur kode internal (*source code*). Setiap poin skenario dirancang untuk memvalidasi satu tujuan akhir guna memastikan integritas fungsional dari setiap modul yang dibangun.

Tabel 4.18 Skenario Pengujian Modul Sentralisasi Multi-Wazuh

ID	Skenario Pengujian (Aksi)	Tujuan Pengujian	Indikator Keberhasilan
F.A.1	Mengisi dan menyimpan formulir penambahan <i>Indexer</i> dengan kredensial yang valid.	Memastikan sistem berhasil memverifikasi koneksi, menyimpan data, dan memicu penarikan riwayat data awal.	Muncul <i>toast notification</i> sukses dan data <i>Indexer</i> baru tampil pada tabel dengan status <i>online</i> .
F.A.2	Menyimpan formulir penambahan <i>Indexer</i> dengan	Memastikan sistem menolak input tidak valid dan	Muncul <i>toast notification</i> berupa pesan <i>error</i> koneksi/otentikasi dan



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

	URL atau kredensial yang salah.	menampilkan pesan <i>error</i> .	data tidak tersimpan ke basis data.
F.A.3	Mengubah sebagian data pada formulir Edit <i>Indexer</i> tanpa menyetel ulang <i>password</i> , lalu menyimpannya.	Memastikan sistem berhasil memperbarui data terkait tanpa merusak <i>password</i> lama yang sudah tersimpan.	Muncul <i>toast notification</i> sukses, pembaruan teks terlihat di antarmuka, dan koneksi tetap berstatus <i>online</i> .
F.A.4	Menghapus <i>Indexer</i> dari daftar dan menyetujui <i>prompt</i> konfirmasi penghapusan.	Memastikan data <i>Indexer</i> terhapus sepenuhnya, baik dari antarmuka visual maupun dari basis data.	Muncul <i>toast notification</i> sukses dan baris data <i>Indexer</i> menghilang dari tabel antarmuka.
F.A.5	Mengisi dan menyimpan formulir penambahan <i>Manager</i> dengan kredensial API yang valid.	Memastikan sistem berhasil memverifikasi koneksi API, mengenkripsi kredensial, dan menyimpan data.	Muncul <i>toast notification</i> sukses dan data <i>Manager</i> baru tampil pada tabel dengan status <i>online</i> .
F.A.6	Menyimpan formulir penambahan <i>Manager</i> dengan	Memastikan sistem mendeteksi kegagalan koneksi API dan mencegah	Muncul <i>toast notification</i> berupa pesan <i>error</i> dan data <i>Manager</i> batal



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

	URL yang tidak terjangkau atau kredensial yang salah.	penyimpanan konfigurasi yang cacat.	ditambahkan ke dalam tabel.
F.A.7	Menghapus <i>Manager</i> dari daftar aktif dan menyetujui <i>prompt</i> konfirmasi penghapusan.	Memastikan data <i>Manager</i> terhapus secara permanen dari antarmuka dan basis data.	Muncul <i>toast notification</i> sukses dan baris data <i>Manager</i> menghilang dari tabel antarmuka.
F.A.8	Mengakses halaman <i>Agents</i> saat minimal terdapat satu <i>Manager</i> yang berstatus <i>online</i> .	Memastikan sistem berhasil memuat daftar agen dan menampilkan total agen aktif secara akurat.	Tabel pada antarmuka terisi dengan baris data agen dan indikator total agen menampilkan angka > 0 .
F.A.9	Menerapkan <i>filter</i> berdasarkan nama <i>Manager</i> tertentu pada menu <i>dropdown</i> .	Memastikan tabel secara dinamis hanya menampilkan agen yang terhubung dengan <i>Manager</i> pilihan.	Seluruh baris agen yang tampil pada tabel hanya memiliki nama <i>Manager</i> yang sesuai dengan parameter <i>filter</i> .
F.A.10	Menerapkan <i>filter</i> berdasarkan status konektivitas	Memastikan tabel secara presisi menyortir agen berdasarkan status	Seluruh baris agen yang tampil pada tabel hanya memiliki label status (misal: warna hijau/ <i>Active</i>) yang sesuai.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

	agen (misal: <i>Active</i>) pada menu <i>dropdown</i> .	yang dipilih pengguna.	
F.A.11	Memasukkan teks berupa nama agen atau alamat IP ke dalam kolom pencarian.	Memastikan sistem langsung memfilter dan menampilkan agen yang relevan tanpa perlu memuat ulang halaman.	Tabel langsung menyusut dan hanya menampilkan agen yang mengandung teks pencarian (IP atau Nama).
F.A.12	Mengaplikasikan parameter <i>filter Manager</i> , <i>filter</i> status, dan kolom pencarian secara bersamaan.	Memastikan logika sistem (<i>AND logic</i>) berfungsi sempurna dalam memproses kriteria <i>filter</i> ganda.	Hasil pada tabel memenuhi ketiga syarat parameter secara bersamaan tanpa anomali data.
F.A.13	Menekan tombol navigasi halaman (<i>Next/Previous</i>) atau memasukkan angka halaman pada kontrol paginasi.	Memastikan sistem mampu memuat dan menampilkan bagian data selanjutnya secara tepat.	Angka halaman pada kontrol paginasi berubah dan tabel memuat baris data agen dari urutan selanjutnya.
F.A.14	Menekan tombol <i>Refresh All Status</i> pada	Memastikan sistem memanggil API seluruh komponen	Muncul <i>toast notification</i> sukses dan indikator status



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

	antarmuka manajemen utama.	Wazuh secara paralel untuk memperbarui status.	komponen (<i>online/offline</i>) diperbarui.
F.A.15	Mengakses halaman <i>Dashboard</i> utama dan melihat <i>area</i> kartu statistik (<i>stat cards</i>).	Memastikan sistem berhasil mengagregasi dan menampilkan total jumlah komponen beserta statusnya.	Angka pada kartu statistik sesuai dengan jumlah aktual <i>Indexer</i> , <i>Manager</i> , dan <i>Agents</i> yang terhubung.

Setelah validasi fungsi sentralisasi *node* Wazuh, pengujian selanjutnya berfokus pada modul integrasi *Threat Intelligence*. Skenario pada Tabel 4.19 dirancang untuk memastikan bahwa sistem mampu mengenali berbagai jenis parameter *Indicator of Compromise* (IoC) dan mengotomatisasi alur investigasi tanpa kendala.

Tabel 4.19 Skenario Pengujian Modul Integrasi Threat Intelligence

ID	Skenario Pengujian (Aksi)	Tujuan Pengujian	Indikator Keberhasilan
F.B.1	Menyimpan API Key <i>AlienVault</i> pada tab pengaturan <i>Intelligence</i> .	Memastikan sistem berhasil memvalidasi kredensial API dan menyimpannya ke dalam basis data.	Muncul <i>toast notification</i> sukses dan API Key tersimpan ke dalam sistem.

**Hak Cipta :**

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

F.B.2	Memasukkan teks <i>Indicator of Compromise</i> (IoC) berformat IPv4 ke dalam kolom pencarian intelijen.	Memastikan sistem secara otomatis mengenali input tersebut sebagai jenis IPv4 dan berhasil menarik data reputasinya.	Antarmuka menampilkan label tipe indikator "IPv4" beserta data geolokasi dan <i>Threat Pulses</i> yang relevan.
F.B.3	Memasukkan teks IoC berformat <i>hash</i> MD5 maupun SHA256 ke dalam kolom pencarian.	Memastikan sistem secara otomatis mengenali input tersebut sebagai jenis <i>file</i> dan berhasil menarik data reputasinya.	Antarmuka menampilkan label tipe indikator " <i>File Hash</i> " beserta detail tipe <i>malware</i> yang relevan.
F.B.4	Memasukkan teks IoC berupa nama domain situs web ke dalam kolom pencarian intelijen.	Memastikan sistem mengidentifikasi format domain dengan tepat dan menyajikan hasil analisis yang sesuai.	Antarmuka menampilkan label tipe indikator " <i>Domain</i> " beserta informasi resolusi DNS terkait.
F.B.5	Menekan tombol pintasan <i>Lookup</i> yang berada tepat di sebelah alamat IP pada daftar log <i>alert</i> di <i>dashboard</i> .	Memastikan sistem secara otomatis memindahkan halaman dan langsung memulai analisis intelijen.	Antarmuka langsung berpindah ke tab <i>Intelligence</i> , kolom pencarian terisi otomatis, dan proses analisis berjalan.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Tahap terakhir dari pengujian fungsional adalah mengevaluasi modul Notifikasi Status Sistem. Mengingat modul ini beroperasi secara asinkron di latar belakang melalui *Docker background scheduler*, pengujiannya tidak hanya berfokus pada ketepatan fungsi, tetapi juga pada kemampuan sistem dalam mendeteksi perubahan status layanan dan mengirimkan notifikasi secara otomatis. Oleh karena itu, Tabel 4.20 menguraikan skenario pengujian yang digunakan untuk memvalidasi mekanisme pemicu (*trigger*) serta rute pengiriman pesan peringatan ke Telegram.

Tabel 4.20 Skenario Pengujian Modul Notifikasi Status Sistem

ID	Skenario Pengujian (Aksi)	Tujuan Pengujian	Indikator Keberhasilan
F.C.1	Menginputkan Bot <i>Token</i> dan <i>Chat ID</i> Telegram yang valid pada formulir pengaturan.	Memastikan sistem dapat memvalidasi dan menyimpan data kredensial bot Telegram dengan aman.	Muncul <i>toast notification</i> sukses dan isian formulir tersimpan di antarmuka.
F.C.2	Menekan tombol <i>Send Test Notification</i> setelah konfigurasi bot Telegram berhasil disimpan.	Memastikan rute komunikasi API berfungsi dengan baik ke perangkat pengguna.	Notifikasi <i>toast</i> sukses muncul di antarmuka web dan pesan uji coba masuk ke aplikasi Telegram pengguna.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

F.C.3	Mengaktifkan sakelar (<i>toggle</i>) pengaturan notifikasi status sistem pada <i>dashboard</i> konfigurasi.	Memastikan sistem menerapkan instruksi untuk mengizinkan pengiriman pesan peringatan operasional.	Posisi <i>toggle</i> berubah menjadi aktif (hijau) dan status tersimpan di basis data.
F.C.4	Mematikan sakelar (<i>toggle</i>) pengaturan notifikasi status sistem pada <i>dashboard</i> konfigurasi.	Memastikan sistem langsung menghentikan pengiriman peringatan otomatis tanpa merusak kredensial bot.	Posisi <i>toggle</i> berubah menjadi non-aktif (abu-abu) dan sistem tidak mengirimkan pesan saat terjadi <i>error</i> .
F.C.5	Memutus koneksi jaringan pada salah satu komponen Wazuh secara sengaja untuk menyimulasikan status <i>offline</i> .	Memastikan sistem mampu mendeteksi kegagalan layanan secara instan dan memicu peringatan (<i>alert</i>).	Teks pesan dengan format " <i>CRITICAL HEALTH ALERT</i> " beserta detail komponen masuk ke aplikasi Telegram.
F.C.6	Memulihkan kembali koneksi jaringan pada komponen Wazuh yang sebelumnya berada dalam status <i>offline</i> .	Memastikan sistem mendeteksi pemulihan tersebut dan memicu pesan pemberitahuan komponen kembali <i>normal</i> .	Teks pesan dengan format " <i>SYSTEM RECOVERED</i> " beserta durasi waktu pemadaman masuk ke aplikasi Telegram.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Setiap poin skenario dirancang untuk memvalidasi satu tujuan akhir guna memastikan integritas fungsional dari setiap modul yang dibangun. Untuk mengkuantifikasi tingkat keberhasilan pada fase ini, keseluruhan hasil pengujian nantinya akan dievaluasi dan dihitung menggunakan Perhitungan Validasi Fungsional (*Black Box Testing*).

B. Prosedur Pengujian Non-Fungsional

Pengujian non-fungsional dirancang untuk memvalidasi kualitas dari sistem yang dibangun. Berbeda dengan pengujian fungsional yang berfokus pada apa yang dilakukan sistem, pengujian ini berfokus pada seberapa baik sistem melakukannya saat dihadapkan pada kondisi dunia nyata. Evaluasi ini krusial untuk membuktikan bahwa Hub Manajemen Terpusat memenuhi standar operasi keamanan yang menuntut kinerja tinggi, keamanan data sensitif yang ketat, serta resiliensi terhadap kegagalan jaringan.

Untuk menghindari anomali data tunggal dan memastikan tingkat akurasi yang tinggi, pengujian kinerja (*performance testing*) pada tahap ini tidak dilakukan melalui observasi sekali jalan. Pengujian mengadopsi prinsip eksperimental kuantitatif yang dilakukan sebanyak 5 (lima) iterasi pada setiap skenario. Pendekatan ini merujuk pada standar pengujian kinerja perangkat lunak oleh Radview (2026) yang menyatakan keharusan untuk menjalankan minimal 5 iterasi *benchmark* di bawah kondisi yang identik.

Pengujian ini menggunakan metode manipulasi variabel terkontrol pada *backend*, *browser developer tools*, terminal, dan *log* jaringan untuk mensimulasikan berbagai kondisi operasional secara konsisten. Pendekatan tersebut memungkinkan setiap skenario diuji dalam kondisi yang terukur sehingga dampaknya terhadap waktu respons dan performa sistem dapat diamati secara objektif. Tabel 4.21 menyajikan rangkaian skenario pengujian beserta hasil yang diharapkan untuk mengevaluasi aspek waktu respons (*response time*) dan performa sistem sesuai dengan kebutuhan non-fungsional poin pertama.



Tabel 4.21 Skenario Pengujian Waktu Respons (Response Time)

ID	Skenario Pengujian (Aksi)	Hasil yang Diharapkan (Expected Result)
NF. 1.1	Waktu Muat Antarmuka: Mengakses Hub Multi-Wazuh sebanyak 5 iterasi. Pengujian menggunakan fitur <i>Network Throttling</i> pada <i>Browser Developer Tools</i> untuk menguji tiga kondisi jaringan: <i>Ideal (No Throttling)</i> , <i>Moderat (4G)</i> , dan <i>Buruk (3G)</i> .	Menemukan batas kondisi jaringan minimum di mana sistem masih dapat melakukan pemuatan halaman dan agregasi data multi-node dengan rata-rata waktu kurang dari 5 detik.
NF. 1.2	Eksekusi Asinkron & Timeout: Menjalankan fitur <i>Lookup Threat Intelligence</i> sebanyak 5 iterasi pada dua kondisi: (A) <i>Fetch Normal</i> , dan (B) <i>Simulasi Server Down</i> menggunakan teknik <i>External API Blackhole</i> (memalsukan alamat API OTX).	Menguji resiliensi <i>thread</i> asinkron untuk memastikan antarmuka tetap responsif (<i>tidak freeze</i>) dan mampu memunculkan <i>Warning Timeout</i> secara konsisten di kisaran batas 15 detik saat terjadi <i>outage</i> pada API pihak ketiga.
NF. 1.3	Latensi Notifikasi: Menyimulasikan kegagalan dan pemulihan komponen dengan mengeksekusi perintah <i>docker compose pause</i> dan <i>unpause</i> pada <i>node Wazuh</i> sebanyak 5 iterasi. Pengujian diulang pada kondisi jaringan <i>normal</i> dan kondisi <i>throttle (3G)</i> .	Mengukur keandalan background <i>scheduler</i> untuk memastikan rata-rata jeda waktu (<i>latency</i>) pengiriman notifikasi ke Telegram tetap berada di bawah ambang batas toleransi 1 menit pada berbagai kondisi jaringan.

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Ketiga skenario pada Tabel 4.21 dirancang untuk mengevaluasi aspek performa sistem dari sudut pandang yang saling melengkapi, yaitu waktu muat antarmuka, responsivitas proses asinkron, dan latensi pengiriman notifikasi. Hasil pengujian dari setiap skenario akan digunakan untuk memverifikasi bahwa sistem tetap memenuhi kebutuhan non-fungsional terkait waktu respons dan performa, baik pada kondisi jaringan normal maupun ketika menghadapi keterbatasan jaringan atau gangguan pada layanan eksternal.

Selanjutnya, Tabel 4.22 menjabarkan daftar skenario pengujian dan hasil yang diharapkan untuk menguji keamanan data (*security*) sesuai kebutuhan non-fungsional poin kedua:

Tabel 4.22 Skenario Pengujian Keamanan Data (Security)

ID	Skenario Pengujian (Aksi)	Hasil yang Diharapkan (<i>Expected Result</i>)
NF2 .1	Enkripsi Database: Mengakses basis data PostgreSQL menggunakan <i>database manager</i> (DBeaver) dan melakukan <i>query</i> ke tabel konfigurasi.	Seluruh <i>field</i> kredensial (<i>Password</i> Wazuh, <i>API Key</i> OTX, <i>Bot Token</i>) tersimpan dalam format <i>ciphertext</i> acak (tidak <i>plaintext</i>).
NF2 .2	Keamanan Otentikasi: Melakukan <i>query</i> ke tabel <i>users</i> di basis data.	Data kata sandi pengguna tersimpan dengan algoritma enkripsi searah (<i>hashing</i>).
NF2 .3	Keamanan Sesi: Melakukan <i>login</i> dan memeriksa <i>tab Application/Cookies</i> pada <i>Browser Developer Tools</i> .	<i>Cookie</i> autentikasi memiliki perlindungan atribut <i>HttpOnly</i> aktif guna mencegah pencurian sesi (<i>XSS mitigation</i>).



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritis atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Terakhir dalam prosedur pengujian non-fungsional, Tabel 4.23 menjabarkan daftar skenario pengujian dan hasil yang diharapkan untuk menguji keandalan (*Reliability & Fault Tolerance*) sesuai kebutuhan non-fungsional poin ketiga:

Tabel 4.23 Skenario Pengujian Keandalan (*Reliability & Fault Tolerance*)

ID	Skenario Pengujian (Aksi)	Hasil yang Diharapkan (<i>Expected Result</i>)
NF 3.1	Toleransi Kegagalan Node Wazuh: Menghubungkan 2 <i>server</i> Wazuh ke Hub, lalu mematikan paksa jaringan (simulasi <i>crash</i>) pada <i>Server A</i> .	Sistem Hub tidak mengalami <i>crash</i> total, dan data agregat dari <i>Server B</i> tetap berhasil ditampilkan secara <i>normal</i> di <i>dashboard</i> , sementara <i>Server A</i> ditandai <i>offline</i> .
NF 3.2	Penanganan Error Integrasi Intelijen: Mengubah konfigurasi API <i>AlienVault OTX</i> menjadi tidak valid (domain yang salah), lalu mengeksekusi fitur Threat Intel <i>Lookup</i> pada sebuah IP.	Sistem tidak mengalami <i>hang</i> atau <i>crash</i> . Antarmuka web memunculkan pesan <i>error</i> melalui <i>toast notification</i> yang rapi, tanpa membocorkan kode <i>backend</i> .
NF 3.3	Ketahanan Modul Notifikasi (Kredensial Tidak Valid): Mengubah Bot <i>Token</i> Telegram menjadi format yang salah/acak pada pengaturan, lalu memutus paksa koneksi salah satu <i>server</i> Wazuh untuk memicu peringatan.	Sistem terjadwal (proses latar belakang) tidak mengalami <i>crash</i> . Pemutusan Wazuh tetap terdeteksi dan diperbarui di <i>dashboard</i> , sementara kegagalan pengiriman pesan Telegram diisolasi secara aman dan dicatat di log <i>backend</i> .



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

C. Prosedur Pengujian Efisiensi Operasional

Pengujian efisiensi bertujuan untuk mendapatkan data kuantitatif terkait optimalisasi alur kerja (*workflow*). Pengujian ini dilakukan dengan menyimulasikan skenario tugas spesifik dan mencatat dua parameter utama: Waktu Penyelesaian (*Time on Task*) dan Jumlah Langkah Interaksi (*Step Count*).

Untuk menjamin akurasi dan objektivitas data, pengukuran waktu tidak lagi mengandalkan pencatatan log *timestamp* secara manual, melainkan menggunakan metode screen recording yang dilengkapi dengan stopwatch digital di layar. Perhitungan waktu dimulai (00:00) sesaat sebelum interaksi pertama (klik/ketik) dilakukan oleh penguji, dan dihentikan tepat ketika sistem menampilkan keluaran (*output*) akhir yang diharapkan. Sementara itu, jumlah langkah dihitung berdasarkan pemetaan interaksi pengguna yang esensial (tindakan membuka/menutup halaman, input teks, dan navigasi halaman) untuk menyelesaikan tugas.

Setiap skenario tugas akan dibandingkan secara head-to-head antara prosedur *baseline* (metode manual/eksisting) dengan prosedur pada sistem usulan (Hub Manajemen Terpusat).

a. Aksi 1: Inventarisasi Komponen Baru

Skenario ini menguji alur kerja saat organisasi menambahkan *server* Wazuh baru ke dalam ekosistem. Pengukuran bertujuan melihat efisiensi pendaftaran *node* baru hingga sistem berhasil memvalidasi koneksi tersebut.

Tabel 4.24 Prosedur Pengujian Efisiensi Aksi 1 (Inventarisasi Komponen Baru)

Tahap	Metode Eksisting (<i>Baseline/Manual</i>)	Metode Sistem Usulan (Hub Terpusat)
Langkah 1	Membuka antarmuka baris perintah (CLI) / Terminal.	Mengakses menu <i>Manage Wazuh</i> pada antarmuka Hub.

**Hak Cipta :**

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Langkah 2	Mengeksekusi SSH ke <i>server</i> sentral / lokal.	Mengisi <i>form</i> penambahan komponen (Nama, URL, Kredensial).
Langkah 3	Membuat atau membuka <i>file</i> teks konfigurasi lokal (misal: menggunakan <i>nano</i>).	Mengklik tombol <i>Save and Test Connection</i> .
Langkah 4	Mengetik dan menyimpan URL, <i>Port</i> , serta kredensial ke dalam <i>file</i> tersebut.	Menerima <i>toast notification</i> yang menyatakan koneksi dan enkripsi berhasil.
Langkah 5	Mengeksekusi perintah <i>curl</i> untuk menguji koneksi secara manual.	-

b. Aksi 2: Pemeliharaan Kredensial (Perubahan *Password*)

Skenario ini menyimulasikan rutinitas rotasi keamanan, di mana administrator harus memperbarui kata sandi *node* Wazuh yang sudah ada. Fokus pengujian adalah melihat kecepatan sistem dalam menerapkan dan memvalidasi kredensial baru.

Tabel 4.25 Prosedur Pengujian Efisiensi Aksi 2 (Pemeliharaan Kredensial)

Tahap	Metode Eksisting (<i>Baseline/Manual</i>)	Metode Sistem Usulan (Hub Terpusat)
Langkah 1	Membuka antarmuka baris perintah (CLI) / Terminal.	Mengakses menu " <i>Manage Wazuh</i> " pada <i>Dashboard</i> Hub.
Langkah 2	Mengakses <i>file</i> penyimpanan kredensial lokal (contoh: <i>nano config.txt</i>).	Mengklik tombol "Edit" pada komponen <i>node</i> yang dituju.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Langkah 3	Menghapus kata sandi lama dan menetikkan kata sandi baru.	Menetikkan kata sandi baru pada kolom <i>Password</i> .
Langkah 4	Menyimpan perubahan pada <i>file</i> lokal.	Mengklik tombol " <i>Update and Test Connection</i> ".
Langkah 5	Mengeksekusi perintah <i>curl</i> untuk memvalidasi autentikasi <i>password</i> baru.	Menerima <i>Toast Notification</i> yang menyatakan pembaruan berhasil.

c. Aksi 3: Pemantauan Kesehatan Multi-Komponen

Skenario ini menguji efisiensi analis dalam memvalidasi ketersediaan layanan (*uptime/availability*) dari klaster Wazuh yang terdiri dari minimal satu *Manager* dan satu *Indexer* dalam kondisi operasional *normal*.

Tabel 4.26 Prosedur Pengujian Efisiensi Aksi 3 (Pemantauan Kesehatan Multi-Komponen)

Tahap	Metode Eksisting (Baseline/Manual)	Metode Sistem Usulan (Hub Terpusat)
Langkah 1	Membuka antarmuka baris perintah (CLI) / Terminal.	Membuka halaman <i>Dashboard</i> utama Hub Manajemen.
Langkah 2	Mengeksekusi perintah <i>curl</i> ke <i>endpoint</i> API Wazuh <i>Indexer</i> (Port 9200).	Membaca status kesehatan seluruh komponen secara visual (Indikator Hijau/Merah) di <i>panel Configured Nodes</i> .
Langkah 3	Membaca respons JSON untuk memastikan <i>Indexer</i> menyala.	-



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Langkah 4	Mengeksekusi perintah curl ke <i>endpoint</i> API Wazuh <i>Manager</i> (Port 55000).	-
Langkah 5	Membaca respons JSON untuk memastikan <i>Manager</i> menyala.	-

d. Aksi 4: Inventarisasi Daftar Agen (*Agent List*)

Skenario ini dilakukan saat administrator perlu memverifikasi daftar seluruh agen yang terhubung ke *server* Wazuh untuk memastikan tidak ada *endpoint* yang terlewat dari pemantauan keamanan.

Tabel 4.27 Prosedur Pengujian Efisiensi Aksi 4 (Inventarisasi Daftar Agen)

Tahap	Metode Eksisting (<i>Baseline/Manual</i>)	Metode Sistem Usulan (Hub Terpusat)
Langkah 1	Membuka antarmuka baris perintah (CLI) / Terminal.	Mengakses <i>tab</i> / menu <i>Wazuh Agents</i> pada aplikasi Hub.
Langkah 2	Mengeksekusi perintah curl autentikasi untuk mendapatkan JSON Web <i>Token</i> (JWT).	Membaca daftar agen dari berbagai <i>server</i> yang telah diintegrasikan ke dalam satu tabel antarmuka.
Langkah 3	Menyalin (<i>copy</i>) <i>string</i> JWT dari respons JSON.	-
Langkah 4	Mengeksekusi perintah curl ke <i>endpoint</i> <i>/agents</i> dengan menyisipkan JWT di <i>header</i> .	-
Langkah 5	Membaca dan memilah (<i>parsing</i>) secara visual	-

**Hak Cipta :**

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

	struktur JSON mentah yang dikembalikan oleh <i>server</i> .	
--	---	--

e. Aksi 5: Investigasi *On-Demand*

Skenario ini menguji efisiensi analis saat melakukan investigasi mendalam terhadap sebuah *Indicator of Compromise* (IoC), seperti alamat IP atau *hash* berkas, yang diperoleh dari tiket insiden eksternal untuk mengetahui status reputasi dan indikasi keterkaitannya dengan aktivitas berbahaya. Pengujian dilakukan untuk mengevaluasi seberapa cepat informasi tersebut dapat diperoleh melalui sistem usulan dibandingkan dengan prosedur investigasi secara manual.

Tabel 4.28 Prosedur Pengujian Efisiensi Aksi 5 (Investigasi *On-Demand*)

Tahap	Metode Eksisting (<i>Baseline/Manual</i>)	Metode Sistem Usulan (Hub Terpusat)
Langkah 1	Membuka <i>tab browser</i> baru dan menavigasi ke situs <i>AlienVault OTX</i> .	Mengakses menu <i>Intelligence</i> pada aplikasi Hub.
Langkah 2	Melakukan <i>login</i> ke akun OTX.	Memasukkan IP/ <i>Hash</i> ke dalam kolom pencarian intelijen dan menekan <i>Enter</i> .
Langkah 3	Menyalin dan menempelkan (<i>paste</i>) IP/ <i>Hash</i> ke kolom pencarian OTX.	Membaca ringkasan intelijen, <i>threat pulse</i> , dan detail forensik yang ditampilkan secara terpadu.
Langkah 4	Menunggu hasil dan menyalin satu per satu informasi reputasi dan <i>threat pulse</i> dari beberapa <i>tab</i> visual yang berbeda.	-

**Hak Cipta :**

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

f. Aksi 6: Investigasi *Alert* (*Click-to-Investigate*)

Skenario ini menguji kecepatan analisis dalam memvalidasi reputasi alamat IP yang dicurigai secara langsung dari log peringatan (*alert*) yang muncul di *dashboard* utama, guna menentukan apakah *alert* tersebut ancaman nyata atau sekadar *false positive*.

Tabel 4.29 Prosedur Pengujian Efisiensi Aksi 6 (Investigasi *Alert*)

Tahap	Metode Eksisting (<i>Baseline/Manual</i>)	Metode Sistem Usulan (Hub Terpusat)
Langkah 1	Menyorot (<i>highlight</i>) dan menyalin alamat IP pada log peringatan.	Mengklik tombol pintasan <i>Lookup</i> di samping alamat IP pada log peringatan.
Langkah 2	Membuka <i>tab browser</i> baru dan menuju situs OTX.	Membaca hasil analisis intelijen yang tersaji secara instan setelah sistem mengarahkan halaman.
Langkah 3	Menempelkan (<i>paste</i>) alamat IP ke kolom pencarian OTX.	-
Langkah 4	Memvalidasi hasil reputasi dari OTX dengan <i>alert</i> asli di log Wazuh secara manual (bolak-balik antar tab).	-

g. Aksi 7: Konfigurasi API *Threat Intelligence*

Skenario ini mensimulasikan langkah administratif yang dilakukan saat mengintegrasikan layanan *Threat Intelligence* eksternal ke dalam ekosistem pemantauan. Tujuannya adalah memastikan sistem dapat berkomunikasi dengan layanan tersebut melalui mekanisme autentikasi yang valid, sehingga setiap permintaan (*query*) dapat diproses dengan aman dan konsisten. Selain itu, konfigurasi ini juga bertujuan memperoleh



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

rate limit yang lebih optimal dibandingkan akses tanpa autentikasi, sehingga proses investigasi dapat berlangsung lebih andal ketika sistem melakukan permintaan ke layanan *Threat Intelligence* secara berulang.

Tabel 4.30 Prosedur Pengujian Efisiensi Aksi 7 (Konfigurasi API Intelijen)

Tahap	Metode Eksisting (<i>Baseline/Manual</i>)	Metode Sistem Usulan (Hub Terpusat)
Langkah 1	Membuka situs <i>AlienVault</i> untuk menyalin <i>API Key</i> dari akun.	Mengakses menu <i>API Configuration</i> pada Hub.
Langkah 2	Membuka terminal/CLI dan mengakses <i>file</i> konfigurasi sistem secara manual.	Memasukkan <i>API Key</i> ke dalam formulir konfigurasi Hub.
Langkah 3	Menempelkan (<i>paste</i>) <i>API Key</i> ke dalam <i>file</i> dan menyimpannya.	Mengklik tombol simpan dan menunggu sistem mengenkripsi kredensial API.
Langkah 4	Melakukan verifikasi integrasi melalui eksekusi skrip di terminal/CLI.	

h. Aksi 8: Konfigurasi Notifikasi

Skenario ini menguji alur kerja administrator saat melakukan pengaturan koneksi antara sistem keamanan dengan platform pesan instan (Telegram) untuk keperluan peringatan dini.

Tabel 4.31 Prosedur Pengujian Efisiensi Aksi 8 (Konfigurasi Notifikasi)

Tahap	Metode Eksisting (<i>Baseline/Manual</i>)	Metode Sistem Usulan (Hub Terpusat)
-------	--	--



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Langkah 1	Membuka terminal/SSH untuk mengakses <i>server</i> target.	Mengakses menu <i>Notifications</i> pada Hub.
Langkah 2	Mencari dan mengedit <i>file</i> skrip notifikasi secara manual.	Memasukkan Bot <i>Token</i> dan <i>Chat ID</i> ke dalam formulir.
Langkah 3	Menyalin <i>token</i> dan ID ke dalam <i>file</i> sistem.	Menekan tombol simpan dan menunggu sistem memvalidasi data.
Langkah 4	Mengatur perizinan (<i>permission</i>) skrip menggunakan perintah <i>chmod/chown</i> .	-
Langkah 5	Mengonfigurasi <i>ossec.conf</i> untuk integrasi dan melakukan <i>restart</i> layanan Wazuh.	-

i. Aksi 9: Uji Coba Integrasi (Connectivity Test)

Skenario ini dilakukan untuk memvalidasi bahwa rute komunikasi kredensial Telegram berfungsi dengan baik sebelum sistem operasional dijalankan sepenuhnya.

Tabel 4.32 Prosedur Pengujian Efisiensi Aksi 9 (Uji Coba Integrasi)

Tahap	Metode Eksisting (Baseline/Manual)	Metode Sistem Usulan (Hub Terpusat)
Langkah 1	Mengakses terminal/SSH dan mencari lokasi skrip integrasi.	Mengklik tombol <i>Send Test Notification</i> pada antarmuka Hub.
Langkah 2	Membuat <i>file</i> JSON <i>dummy</i> secara manual untuk	Menerima <i>toast notification</i> di layar web



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

	menyimulasikan data <i>alert</i> uji coba.	dan mengecek pesan yang masuk di aplikasi Telegram.
Langkah 3	Mengeksekusi skrip melalui baris perintah (CLI).	-
Langkah 4	Mengecek aplikasi Telegram secara terpisah untuk melihat hasil.	-

j. Aksi 10: Pemantauan Status Komponen (Insiden Down)

Skenario ini membandingkan beban kerja analis dalam menemukan dan mengidentifikasi komponen yang mati. Pada metode eksisting, identifikasi sangat bergantung pada rutinitas pengecekan manual secara berkala (*Pull mechanism*), sehingga analis harus mencari tahu sendiri status *server*. Sebaliknya, sistem usulan memanfaatkan background *scheduler* (*Sentinel*) untuk mendeteksi dan mengirimkan *diagnosis* secara proaktif (*Push mechanism*).

Tabel 4.33 Prosedur Pengujian Efisiensi Aksi 10 (Pemantauan Status)

Tahap	Metode Eksisting (<i>Baseline/Manual</i>)	Metode Sistem Usulan (<i>Hub Terpusat</i>)
Prasyarat (Langkah 0)	Mematikan layanan salah satu komponen Wazuh target secara paksa (misal: <code>docker compose pause wazuh.indexer</code>) untuk menyimulasikan <i>downtime</i> .	
Langkah 1	Membuka catatan inventaris/dokumen (berisi daftar IP dan Port) untuk melihat target komponen yang akan dicek rutin.	Menerima notifikasi <i>push</i> di perangkat dan membuka aplikasi Telegram.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Langkah 2	Membuka terminal (CLI) dan merakit perintah pengujian (misal: <i>curl</i> atau <i>ping</i>) berdasarkan data IP target.	Membaca rangkuman pesan CRITICAL HEALTH ALERT yang telah memuat identitas komponen dan jenis kegagalan (<i>error details</i>).
Langkah 3	Mengeksekusi perintah tersebut dan menunggu hingga jaringan mencapai batas <i>timeout</i> .	-
Langkah 4	Membaca dan menafsirkan pesan <i>error</i> mentah pada terminal (misal: <i>Connection refused</i> atau <i>Timeout</i>) untuk mengonfirmasi kerusakan.	-

4.4.3 Data Hasil Pengujian

Subbab ini memaparkan seluruh data empiris yang diperoleh dari pelaksanaan prosedur pengujian yang telah dirancang pada tahap sebelumnya. Hasil pengujian disajikan secara sistematis agar setiap aspek kinerja sistem dapat dievaluasi berdasarkan data yang diperoleh selama proses pengujian. Penyajian hasil ini bertujuan menunjukkan sejauh mana sistem mampu memenuhi tujuan penelitian serta kebutuhan fungsional dan non-fungsional yang telah ditetapkan. Pembahasan dibagi ke dalam tiga bagian utama, yaitu: (A) Hasil Pengujian Fungsional untuk memastikan setiap fitur bekerja sesuai dengan rancangan dan tidak mengalami kecacatan logika, (B) Hasil Pengujian Non-Fungsional untuk mengevaluasi atribut kualitas, performa, dan keandalan sistem, serta (C) Hasil Pengujian Efisiensi Operasional untuk mengukur dampak implementasi sistem terhadap pengurangan beban kerja dan penyederhanaan proses administrasi secara kuantitatif.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

A. Data Hasil Pengujian Fungsional (*Black Box Testing*)

Pengujian fungsional dieksekusi secara menyeluruh terhadap 26 skenario pengujian yang mencakup modul Hub Multi-Wazuh, integrasi *Threat Intelligence*, dan sistem Notifikasi *Real-Time*. Indikator keberhasilan diukur berdasarkan kesesuaian antara *expected result* dengan respons aktual sistem (*actual result*). Berikut adalah rekapitulasi data hasil pengujian fungsional.

Tabel 4.34 Hasil Pengujian Modul Sentralisasi Multi-Wazuh

ID	Skenario Pengujian (Aksi)	Hasil Aktual (<i>Actual Result</i>)	Status
F.A.1	Mengisi dan menyimpan formulir penambahan <i>Indexer</i> dengan kredensial yang valid.	Sistem menampilkan <i>toast notification</i> sukses dan data <i>Indexer</i> baru tampil di tabel dengan status <i>online</i> .	Berhasil
F.A.2	Menyimpan formulir penambahan <i>Indexer</i> dengan URL atau kredensial yang salah.	Sistem menampilkan pesan <i>error</i> koneksi/autentikasi dan data tidak tersimpan.	Berhasil
F.A.3	Mengubah sebagian data pada formulir Edit <i>Indexer</i> tanpa menyetel ulang <i>password</i> .	Pembaruan teks terlihat di antarmuka, <i>toast notification</i> muncul, dan status tetap <i>online</i> .	Berhasil
F.A.4	Menghapus <i>Indexer</i> dari daftar dan menyetujui <i>prompt</i> konfirmasi penghapusan.	<i>Toast notification</i> sukses muncul dan baris data <i>Indexer</i> menghilang dari tabel.	Berhasil
F.A.5	Mengisi dan menyimpan formulir penambahan	Sistem menampilkan <i>toast notification</i> sukses dan data	Berhasil



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

	<i>Manager</i> dengan kredensial API yang valid.	<i>Manager</i> baru tampil di tabel dengan status <i>online</i> .	
F.A.6	Menyimpan formulir penambahan <i>Manager</i> dengan URL yang tidak terjangkau.	Sistem menampilkan <i>toast notification</i> berisi pesan <i>error</i> dan data batal ditambahkan.	Berhasil
F.A.7	Menghapus <i>Manager</i> dari daftar aktif dan menyetujui <i>prompt</i> konfirmasi.	<i>Toast notification</i> sukses muncul dan baris data <i>Manager</i> menghilang dari tabel.	Berhasil
F.A.8	Mengakses halaman <i>Agents</i> saat minimal terdapat satu <i>Manager</i> yang berstatus <i>online</i> .	Tabel terisi penuh dengan baris data agen dari <i>server</i> dan total agen dikalkulasi dengan benar.	Berhasil
F.A.9	Menerapkan <i>filter</i> berdasarkan nama <i>Manager</i> tertentu pada menu <i>dropdown</i> .	Tabel merender ulang dan secara akurat hanya menampilkan agen dari <i>Manager</i> yang dipilih.	Berhasil
F.A.10	Menerapkan <i>filter</i> berdasarkan status konektivitas agen pada menu <i>dropdown</i> .	Tabel secara akurat hanya menampilkan agen dengan indikator warna status yang diminta.	Berhasil
F.A.11	Memasukkan teks berupa nama agen atau alamat IP ke dalam kolom pencarian.	<i>Delay typing (debounce)</i> berfungsi, lalu tabel langsung menampilkan hasil pencarian yang tepat.	Berhasil
F.A.12	Mengaplikasikan parameter <i>filter Manager</i> , <i>filter</i> status,	Data yang tampil di tabel tervalidasi memenuhi ketiga kondisi <i>filter</i> tanpa adanya <i>error</i> .	Berhasil



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

	dan kolom pencarian bersamaan.		
F.A.13	Menekan tombol navigasi halaman (<i>Next/Previous</i>) pada kontrol paginasi tabel.	Transisi halaman berjalan mulus dan tabel memuat <i>batch</i> data agen berikutnya dengan akurat.	Berhasil
F.A.14	Menekan tombol <i>Refresh All Status</i> pada antarmuka manajemen utama.	Indikator <i>toast notification</i> muncul dan status seluruh koneksi diperbarui ke kondisi terkini.	Berhasil
F.A.15	Mengakses halaman <i>Dashboard</i> utama dan melihat <i>area</i> kartu statistik (<i>stat cards</i>).	Angka pada <i>stat cards</i> tervalidasi sinkron dengan data <i>backend</i> tanpa adanya perbedaan jumlah.	Berhasil

Setelah prosedur modul sentralisasi, pengujian dilanjutkan ke modul pengayaan informasi (*Threat Intelligence*). Evaluasi ini bertujuan untuk membuktikan bahwa sistem mampu merespons berbagai tipe indikator ancaman secara dinamis serta mengeliminasi proses *copy-paste* manual yang membebani analis.

Tabel 4.35 Hasil Pengujian Modul Sentralisasi Threat Intelligence

ID	Skenario Pengujian (Aksi)	Hasil Aktual (<i>Actual Result</i>)	Status
F.B.1	Menyimpan API Key <i>AlienVault</i> pada tab pengaturan <i>Intelligence</i> .	<i>Toast notification</i> sukses muncul dan kredensial API tervalidasi serta tersimpan dalam sistem.	Berhasil
F.B.2	Memasukkan teks <i>Indicator of Compromise</i> (IoC)	Sistem secara presisi merender data intelijen dengan label tipe indikator "IPv4".	Berhasil

**Hak Cipta :**

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

	berformat IPv4 ke dalam kolom pencarian.		
F.B.3	Memasukkan teks IoC berformat <i>hash</i> MD5 maupun SHA256 ke dalam kolom pencarian.	Sistem secara presisi merender data intelijen dengan label tipe indikator " <i>File Hash</i> ".	Berhasil
F.B.4	Memasukkan teks IoC berupa nama domain situs web ke dalam kolom pencarian intelijen.	Sistem secara presisi merender data intelijen dengan label tipe indikator " <i>Domain</i> ".	Berhasil
F.B.5	Menekan tombol pintasan <i>Lookup</i> di sebelah alamat IP pada daftar log <i>alert</i> di <i>dashboard</i> .	Antarmuka langsung berpindah halaman, parameter terisi otomatis, dan memicu penarikan data secara instan.	Berhasil

Rangkaian pengujian fungsional diakhiri dengan evaluasi pada modul Notifikasi Status Sistem. Pengujian ini sangat krusial untuk memastikan bahwa keandalan proses latar belakang (*background scheduler*) terhubung sempurna dengan API Telegram dan tidak menghasilkan *spam* notifikasi.

Tabel 4.36 Hasil Pengujian Modul Telegram Notifikasi Status Sistem

ID	Skenario Pengujian (Aksi)	Hasil Aktual (<i>Actual Result</i>)	Status
F.C.1	Menginputkan Bot <i>Token</i> dan <i>Chat ID</i> Telegram yang valid pada formulir pengaturan.	<i>Toast notification</i> sukses muncul dan konfigurasi bot tersimpan di dalam sistem.	Berhasil



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

F.C.2	Menekan tombol <i>Send Test Notification</i> setelah konfigurasi bot Telegram disimpan.	Antarmuka web menampilkan notifikasi sukses dan pesan uji coba diterima di aplikasi Telegram.	Berhasil
F.C.3	Mengaktifkan sakelar (<i>toggle</i>) pengaturan notifikasi status sistem pada <i>dashboard</i> .	Posisi <i>toggle</i> tersimpan aktif dan <i>scheduler</i> mulai mendengarkan perubahan status <i>node</i> .	Berhasil
F.C.4	Mematikan sakelar (<i>toggle</i>) pengaturan notifikasi status sistem pada <i>dashboard</i> .	Posisi <i>toggle</i> tersimpan non-aktif dan <i>scheduler</i> menghentikan rute pengiriman pesan.	Berhasil
F.C.5	Memutus koneksi jaringan pada komponen Wazuh secara sengaja (simulasi <i>offline</i>).	Sistem sukses mengirimkan pesan " <i>CRITICAL HEALTH ALERT</i> " yang terformat rapi ke Telegram.	Berhasil
F.C.6	Memulihkan kembali koneksi jaringan pada komponen Wazuh yang sebelumnya <i>offline</i> .	Sistem sukses mengirimkan pesan " <i>SYSTEM RECOVERED</i> " beserta kalkulasi durasi ke Telegram.	Berhasil

Berdasarkan pemaparan data pada Tabel 4.34, Tabel 4.35, dan Tabel 4.36, seluruh skenario yang diujikan berjalan sesuai dengan rancangan spesifikasi sistem. Dari total 26 *test case* yang dieksekusi, baik pada skenario penginputan data yang valid (*happy path*) maupun penanganan *error* saat terjadi kegagalan



(*unhappy path*) tidak ditemukan adanya anomali, *bug*, atau *system crash* dan semuanya memenuhi indikator keberhasilan (berhasil).

B. Data Hasil Pengujian Non-Fungsional

Pengujian non-fungsional dilakukan untuk memvalidasi atribut kualitas arsitektur sistem dalam skenario operasional dunia nyata (*real-world deployment*). Pengujian ini dibagi menjadi tiga aspek utama: kinerja (*performance*), keamanan (*security*), dan keandalan (*reliability*). Khusus untuk pengujian kinerja, evaluasi dilakukan menggunakan metode eksperimental dengan 5 (lima) iterasi pada setiap skenario untuk mendapatkan nilai rata-rata yang empiris.

Pengujian Kinerja Waktu Muat Antarmuka (NF.1.1) bertujuan untuk menemukan batas kondisi jaringan minimum di mana sistem masih mampu memuat Hub Multi-Wazuh dan mengagregasi data multi-*node*. Pengujian ini dilakukan dengan memanipulasi kecepatan jaringan menggunakan fitur *Network Throttling* pada Browser Developer Tools.

Tabel 4.37 Hasil Pengujian Waktu Muat Antarmuka (NF.1.1)

Iterasi	Waktu Muat Kondisi <i>Ideal</i> (<i>No Throttling</i>)	Waktu Muat Kondisi <i>Moderat</i> (<i>Fast 4G</i>)	Waktu Muat Kondisi <i>Buruk</i> (<i>Slow 3G</i>)
1	2,27 s	6,23 s	40,36 s
2	2,96 s	9,78 s	38,72 s
3	2,79 s	4,87 s	38,89 s
4	1,80 s	7,01 s	34,27 s
5	2,28 s	3,54 s	30,55 s
Rata-rata	2,42 s	6,28 s	36,55 s

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Berdasarkan Tabel 4.37, perolehan data pada kondisi jaringan *Ideal* mencatat rata-rata waktu muat sebesar 2,42 detik. Pada kondisi *Moderat* (4G), rata-rata waktu muat tercatat sebesar 6,28 detik, sedangkan pada kondisi jaringan *Buruk* (3G) rata-rata waktu muat mencapai 36,55 detik.

Pengujian Kinerja Eksekusi Asinkron dan *Timeout* (NF.1.2) bertujuan untuk menguji resiliensi *thread* asinkron saat menarik data *Threat Intelligence* dari API eksternal. Skenario (A) menguji penarikan data *normal*, sementara Skenario (B) menyuntikkan kegagalan buatan menggunakan teknik External API *Blackhole* (yakni dengan memalsukan *link* API OTX) untuk memastikan sistem memutus koneksi secara paksa tanpa membekukan (*freeze*) antarmuka pengguna dengan hasil dirangkum pada tabel 4.38.

Tabel 4.38 Hasil Pengujian Eksekusi Asinkron & Timeout (NF.1.2)

Iterasi	Case A: Kondisi Normal		Case B: Kondisi External API Blackhole	
	Waktu	UI Timeout Warning	Waktu	UI Timeout Warning
1	11,2s	Tidak	16,13 s	Ya
2	6,95 s	Tidak	22,89 s	Ya
3	6,34 s	Tidak	16,28 s	Ya
4	8,00 s	Tidak	18,16 s	Ya
5	4,09 s	Tidak	16,01 s	Ya
Rata-rata	7,30 s		17,89 s	

**Hak Cipta :**

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Tabel 4.38 memaparkan perbandingan waktu eksekusi dari kedua kondisi tersebut. Pada *Case A*, rata-rata waktu yang tercatat adalah 7,30 detik tanpa memunculkan peringatan batas waktu. Pada *Case B*, sistem mencatat rata-rata waktu sebesar 17,89 detik dengan kemunculan *Warning Timeout* secara konsisten pada seluruh iterasi, disertai status antarmuka yang tetap interaktif (tidak *freeze*).

Pengujian Latensi Notifikasi *Real-Time* (NF.1.3) dilakukan untuk mengukur jeda waktu (*delay*) dari background *scheduler* dalam mendeteksi dan mengirimkan peringatan saat terjadi anomali pada *node* infrastruktur. Pengujian dilakukan dengan memutus (*pause*) koneksi *container* secara paksa, lalu menghitung selisih waktu antara log *backend* yang mencatat kegagalan dengan waktu masuknya pesan di Telegram dengan hasil dirangkum pada tabel 4.39.

Tabel 4.39 Hasil Pengujian Latensi Notifikasi (NF.1.3)

Iterasi	<i>Case A (Normal) : Tanpa Throttle</i>			<i>Case B: Throttling 3G</i>		
	Waktu Komponen Down/Up	Waktu Telegram	Selisih Waktu (s)	Waktu Komponen Down/Up	Waktu Telegram	Selisih Waktu (s)
1	11:13:57	11:14:44	47	11:35:05	11:35:44	39
2	11:18:33	11:18:50	17	11:37:03	11:37:45	42
3	11:19:15	11:19:45	30	11:38:13	11:38:49	36
4	11:24:12	11:24:49	37	11:48:59	11:49:49	50
5	11:25:10	11:25:45	35	11:58:50	11:59:49	59
Rata-rata	33.2s			45.2s		

Merujuk pada Tabel 4.39, pengujian pada *Case A* (Jaringan *Normal*) menghasilkan selisih waktu rata-rata sebesar 33,2 detik antara pencatatan log


Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

backend dengan penerimaan pesan di Telegram. Pada *Case B* dengan kondisi jaringan yang dibatasi (*Throttling 3G*), rata-rata latensi pengiriman pesan tercatat sebesar 45,2 detik.

Pengujian pada aspek keamanan data (NF 2.1-2.3) difokuskan pada observasi status penyimpanan kredensial dan perlindungan sesi pengguna di dalam sistem. Pengujian ini tidak melibatkan iterasi karena bersifat validasi kondisi absolut (*binary state*) pada tingkat basis data pengaturan browser.

Tabel 4.40 Hasil Pengujian Keamanan Data (NF.2)

ID	Skenario Pengujian (Aksi)	Hasil Aktual (<i>Actual Result</i>)
NF. 2.1	Mengakses tabel konfigurasi pada basis data PostgreSQL.	Kolom <code>_encrypted_password</code> tervalidasi menyimpan data dalam format <i>ciphertext</i> AES (acak).
NF. 2.2	Melakukan <i>query</i> ke tabel pengguna (<i>users</i>) di basis data utama.	Kolom <i>password_hash</i> tervalidasi menyimpan kata sandi dalam format <i>hashing</i> (script:).
NF. 2.3	Memeriksa tab <i>Application/Cookies</i> pada <i>Browser Developer Tools</i> .	<i>Cookie</i> sesi tervalidasi memiliki atribut <i>HttpOnly</i> dan <i>Secure</i> dalam keadaan aktif (tercentang).

Berdasarkan hasil observasi empiris pada pengujian enkripsi basis data (NF.2.1) yang ditunjukkan pada Lampiran 9, eksekusi *query* ke dalam tabel *wazuh_indexers* memperlihatkan bahwa data kredensial tidak tersimpan dalam format teks terang (*plaintext*). Kondisi serupa juga ditemukan pada pengujian keamanan autentikasi pengguna (NF.2.2) di Lampiran 10, di mana kata sandi pengguna telah diacak menggunakan algoritma hashing. Selanjutnya, pada pengujian keamanan sesi (NF.2.3), hasil observasi pada Lampiran 11 membuktikan bahwa *cookie* yang digunakan untuk autentikasi antarmuka web telah terproteksi secara memadai guna memitigasi pencurian sesi.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Terakhir, pengujian keandalan (NF.3.1-NF3.3) dilakukan untuk memvalidasi respons sistem saat dihadapkan pada anomali jaringan, kegagalan *node*, maupun *error* pada API pihak ketiga. Pengujian ini bersifat observasional terhadap respons antarmuka dan pencatatan log sistem.

Tabel 4.41 Hasil Pengujian Keandalan (*Reliability*)

ID	Skenario Pengujian (Aksi)	Hasil Aktual (<i>Actual Result</i>)
NF.3 .1	Mematikan paksa jaringan pada salah satu <i>server</i> Wazuh.	<i>Dashboard</i> tetap memuat <i>node</i> lain secara <i>normal</i> , dan <i>node</i> yang mati dilabeli warna merah (<i>Disconnected</i>).
NF.3 .2	Memalsukan <i>link</i> API <i>AlienVault</i> OTX dan mengeksekusi fitur <i>Lookup</i> .	Antarmuka web tidak <i>crash</i> (<i>Error</i> 500), melainkan memunculkan <i>toast notification</i> berisi pesan gagal.
NF.3 .3	Memalsukan Bot <i>Token</i> Telegram dan memicu peringatan <i>node offline</i> .	Kegagalan API Telegram tercatat di log <i>backend</i> , namun proses <i>background scheduler</i> tetap berjalan <i>normal</i> .

Pada pengujian toleransi kegagalan *node* (NF.3.1), simulasi pemutusan koneksi pada salah satu *server* Wazuh menghasilkan duyung visual seperti yang tertera pada Lampiran 12. Antarmuka *dashboard* tetap beroperasi tanpa *crash* atau layar kosong (*blank screen*). Pengujian penanganan *error* integrasi intelijen (NF.3.2) yang ditunjukkan pada Lampiran 13 juga memperlihatkan bahwa kegagalan resolusi API di tingkat *backend* berhasil ditangkap dan diisolasi oleh sistem menjadi *toast notification* yang rapi di layar pengguna.

Terakhir, pengujian ketahanan modul notifikasi (NF.3.3) yang dilakukan dengan menyuntikkan Bot *Token* Telegram tidak valid (Lampiran 14) memunculkan peringatan kegagalan di antarmuka. Lebih lanjut, observasi pada log *backend* di Lampiran 15 menunjukkan bahwa meskipun sistem



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

gagal melakukan HTTP *POST* ke Telegram, siklus background *scheduler* tetap melanjutkan eksekusinya tanpa mengalami penghentian proses secara *fatal*.

C. Data Hasil Pengujian Efisiensi Operasional

Pengujian efisiensi operasional telah dieksekusi dengan menyimulasikan 10 (sepuluh) skenario aksi operasional (*workflow*) dari sudut pandang analisis keamanan. Pengukuran dilakukan dengan mengekstraksi data *timestamp* rekaman layar saat pengujian berlangsung untuk mendapatkan durasi waktu aktual, serta memetakan jumlah langkah interaksi yang dilakukan oleh pengguna pada setiap skenario.

Data hasil pengujian ini dikelompokkan ke dalam dua parameter ukur utama, yaitu Waktu Penyelesaian (*Time on Task*) dan Jumlah Langkah Interaksi (*Step Count*), yang masing-masing dibandingkan antara metode eksisting (manual) dan metode sistem usulan (Hub Terpusat). Tabel 4.42 menyajikan rekapitulasi data durasi waktu penyelesaian untuk masing-masing skenario pengujian dalam satuan detik.

Tabel 4.42 Rekapitulasi Data Waktu Penyelesaian (*Time on Task*)

ID Aksi	Deskripsi Skenario Operasional	Waktu Metode Eksisting (<i>Baseline/Manual</i>)	Waktu Metode Usulan (Hub Terpusat)
Aksi 1	Inventarisasi Komponen Baru	63 detik	37 detik
Aksi 2	Pemeliharaan Kredensial	74 detik	25 detik
Aksi 3	Pemantauan Kesehatan Multi-Komponen	58 detik	20 detik



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Aksi 4	Inventarisasi Daftar Agen (<i>Agent List</i>)	42 detik	12 detik
Aksi 5	Investigasi <i>On-Demand</i> (OTX)	25 detik	17 detik
Aksi 6	Investigasi <i>Alert</i> (<i>Click-to-Investigate</i>)	38 detik	20 detik
Aksi 7	Konfigurasi API <i>Threat Intelligence</i>	39 detik	14 detik
Aksi 8	Konfigurasi Notifikasi (Telegram)	66 detik	19 detik
Aksi 9	Uji Coba Integrasi (<i>Connectivity Test</i>)	44 detik	25 detik
Aksi 10	Identifikasi Akar Masalah Insiden <i>Down</i>	67 detik	26 detik

Selain waktu penyelesaian, efisiensi operasional juga dinilai dari seberapa ringkas interaksi pengguna (*user interaction*) yang dibutuhkan untuk menyelesaikan sebuah tugas. Tabel 4.43 menyajikan rekapitulasi perhitungan jumlah langkah (*step count*) berdasarkan prosedur eksekusi yang telah diuraikan pada tahap desain pengujian, dengan hanya menghitung jumlah langkah yang dilakukan user dan bukan sistem.

Tabel 4.43 Rekapitulasi Data Jumlah Langkah Interaksi (Step Count)

ID Aksi	Deskripsi Skenario Operasional	Langkah Metode Eksisting (<i>Baseline/Manual</i>)	Langkah Metode Usulan (Hub Terpusat)
Aksi 1	Inventarisasi Komponen Baru	6 Langkah	3 Langkah



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Aksi 2	Pemeliharaan Kredensial	4 Langkah	3 Langkah
Aksi 3	Pemantauan Kesehatan Multi-Komponen	5 Langkah	2 Langkah
Aksi 4	Inventarisasi Daftar Agen (<i>Agent List</i>)	5 Langkah	1 Langkah
Aksi 5	Investigasi <i>On-Demand</i> (OTX)	6 Langkah	4 Langkah
Aksi 6	Investigasi <i>Alert</i> (<i>Click-to-Investigate</i>)	5 Langkah	2 Langkah
Aksi 7	Konfigurasi API <i>Threat Intelligence</i>	4 Langkah	3 Langkah
Aksi 8	Konfigurasi Notifikasi (Telegram)	6 Langkah	3 Langkah
Aksi 9	Uji Coba Integrasi (<i>Connectivity Test</i>)	5 Langkah	2 Langkah
Aksi 10	Identifikasi Akar Masalah Insiden <i>Down</i>	4 Langkah	1 Langkah

Seluruh data empiris yang telah dikumpulkan dan dipaparkan pada Tabel 4.42 dan Tabel 4.43 ini selanjutnya akan digunakan sebagai basis perhitungan komparatif. Penjabaran persentase tingkat efisiensi dan interpretasi dampaknya terhadap operasional Security Operations Center (SOC) dibahas secara komprehensif pada subbab evaluasi pengujian.

4.4.4 Analisis Data / Evaluasi Pengujian

Subbab ini menyajikan interpretasi dan evaluasi komprehensif terhadap data empiris yang telah dipaparkan pada tahap hasil pengujian. Analisis dilakukan

**Hak Cipta :**

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritis atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

untuk mengukur sejauh mana sistem yang dibangun berhasil memenuhi standar operasional dan menjawab rumusan masalah yang telah ditetapkan. Evaluasi pada bagian ini ditinjau dari pencapaian fungsionalitas sistem serta pemenuhan atribut kualitas non-fungsional.

A. Analisis Pengujian Fungsional (*Black Box Testing*)

Berdasarkan hasil pengujian fungsional, seluruh skenario yang telah dirancang dapat dijalankan sesuai dengan keluaran yang diharapkan (*expected result*). Pengujian mencakup 26 skenario yang merepresentasikan fungsi-fungsi utama sistem, meliputi Modul Sentralisasi Multi-Wazuh, Modul Sentralisasi *Threat Intelligence*, dan Modul Telegram Notifikasi Status Sistem. Rekapitulasi hasil pengujian menunjukkan hasil seperti berikut:

- i) Modul Sentralisasi Multi-Wazuh: 15 dari 15 skenario berhasil.
- ii) Modul Sentralisasi *Threat Intelligence*: 5 dari 5 skenario berhasil.
- iii) Modul Telegram Notifikasi Status Sistem: 6 dari 6 skenario berhasil.

Dengan demikian, seluruh 26 skenario yang dirancang pada tahap pengujian dapat dijalankan sesuai dengan hasil yang diharapkan pada ruang lingkup penelitian ini.

Berdasarkan hasil tersebut, terdapat beberapa temuan yang dapat disimpulkan. Pertama, seluruh skenario pada alur operasional utama (*happy path*) menunjukkan bahwa integrasi antara Wazuh, AlienVault OTX, dan Telegram telah berjalan sesuai dengan kebutuhan sistem. Hub Manajemen Terpusat mampu mengintegrasikan ketiga layanan tersebut ke dalam satu antarmuka sehingga proses administrasi, investigasi, dan pemantauan dapat dilakukan tanpa harus berpindah antarplatform.

Kedua, skenario yang menguji kondisi kesalahan (*unhappy path*), seperti kredensial yang tidak valid, *API Key* yang salah, maupun alamat server yang tidak dapat dijangkau, menunjukkan bahwa sistem mampu memberikan respons yang sesuai terhadap masukan yang tidak valid. Selain itu, mekanisme *Validate Before Save* berfungsi sebagaimana dirancang dengan



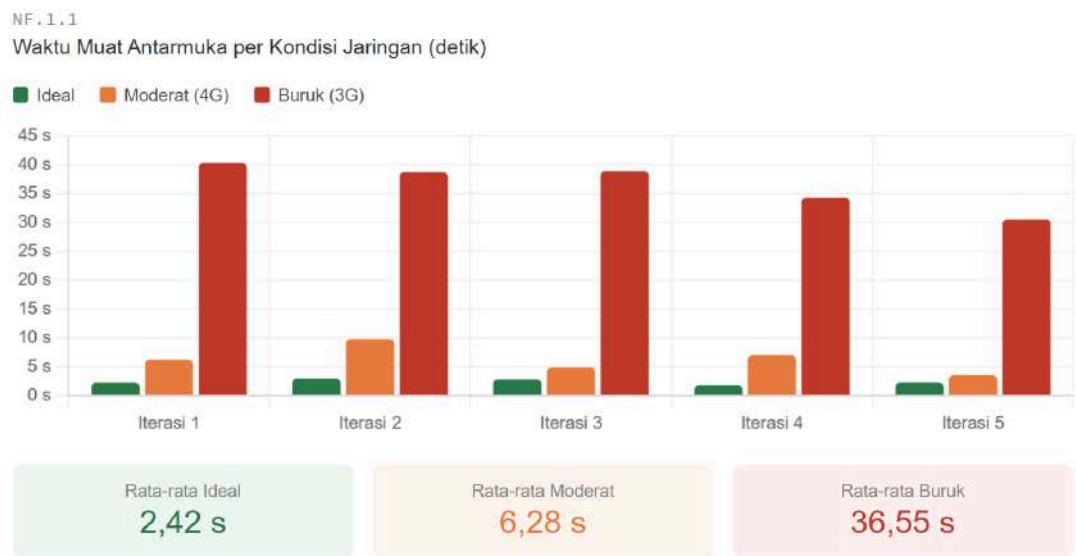
Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

mencegah konfigurasi yang tidak valid tersimpan ke dalam basis data selama proses pengujian. Temuan ini menunjukkan bahwa mekanisme validasi yang diimplementasikan telah mendukung konsistensi konfigurasi sistem pada skenario yang diuji.

B. Analisis Pengujian Non-Fungsional

Berbeda dengan pengujian fungsional yang berfokus pada validitas logika, analisis non-fungsional ditujukan untuk mengukur sejauh mana arsitektur sistem mampu bertahan dan beroperasi di bawah kondisi *real-world* yang fluktuatif. Parameter keberhasilan pada fase ini tidak lagi diukur secara biner (berhasil/gagal), melainkan dikomparasikan secara kuantitatif terhadap standar dan ambang batas (*threshold*) literatur operasional. Analisis pertama berfokus pada kinerja waktu muat antarmuka (NF.1.1) dengan hasil temuan divisualisasikan pada Gambar 4.87.



Gambar 4.87 Grafik Komparasi Waktu Muat Antarmuka

Merujuk pada standar toleransi kognitif yang dikemukakan oleh Kelton et al. (2017), pemuatan objek visual utama harus diselesaikan di bawah 5 detik agar fokus kognitif analis tidak terputus (*User-Perceived Page Load Time*). Berdasarkan grafik tersebut, sistem sangat responsif pada kondisi jaringan Ideal dengan waktu muat rata-rata 2,42 detik. Kinerja cepat ini merupakan hasil langsung dari penerapan *caching* Redis di sisi *backend*. Pada kondisi



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Moderat (4G), waktu muat sedikit naik menjadi 6,28 detik. Namun, pada kondisi Buruk (3G), waktu melonjak drastis hingga 36,55 detik.

Data ini menunjukkan bahwa meskipun *backend* sudah dioptimalkan, performa sistem tetap bergantung pada kualitas koneksi internet. Kesimpulannya, agar analisis SOC dapat bekerja dengan efisien sesuai standar waktu 5 detik, implementasi Hub Manajemen Terpusat ini membutuhkan infrastruktur jaringan yang *ideal*, ataupun setidaknya 4G yang masih berada dalam rentang toleransi batas 5 detik tersebut.

Bergerak lebih dalam ke lapisan *backend*, analisis kinerja eksekusi asinkron (NF.1.2) divisualisasikan pada Gambar 4.88 memvalidasi ketahanan sistem terhadap kegagalan koneksi API pihak ketiga.



Gambar 4.88 Grafik Komparasi Waktu Eksekusi Asinkron dan *Timeout*

Grafik pada Gambar 4.84 menunjukkan bahwa proses penarikan data pada kondisi normal (Case A) memerlukan waktu rata-rata 7,30 detik. Ketika layanan eksternal disimulasikan tidak dapat diakses (Case B), sistem secara konsisten menghentikan proses pada rata-rata 17,89 detik. Nilai tersebut masih sesuai dengan rancangan sistem karena merupakan akumulasi dari batas *timeout* 15 detik dan waktu yang diperlukan untuk memproses serta mengembalikan respons kepada pengguna. Selama proses tersebut berlangsung, antarmuka tetap responsif dan tidak mengalami *freeze*, sehingga



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

pengguna masih dapat berinteraksi dengan aplikasi meskipun layanan pihak ketiga sedang mengalami gangguan.

Hasil ini menunjukkan bahwa mekanisme *timeout* dan pemrosesan asinkron yang diterapkan mampu mencegah permintaan yang menggantung (*hanging request*) menghambat kinerja aplikasi. Temuan tersebut sejalan dengan Velamoor (2024), yang menjelaskan bahwa penerapan *timeout* pada koneksi API penting untuk membatasi waktu tunggu terhadap layanan eksternal, sehingga sumber daya server tidak terus terpakai oleh permintaan yang tidak kunjung memperoleh respons.

Pada aspek operasional proaktif (NF.1.3), seperti hasil yang ditunjukkan pada Gambar 4.85, kecepatan pengiriman notifikasi menjadi faktor penting karena menentukan seberapa cepat administrator dapat mengetahui dan merespons gangguan yang terjadi.



Gambar 4.89 Grafik Komparasi Waktu Latensi Notifikasi

Hasil pengujian pada Gambar 4.89 menunjukkan bahwa mekanisme pengiriman notifikasi mampu beroperasi secara konsisten pada berbagai kondisi jaringan. Pada kondisi jaringan normal, rata-rata waktu yang dibutuhkan sistem untuk mengirimkan notifikasi ke Telegram adalah 33,2 detik. Ketika kondisi jaringan diturunkan hingga setara 3G, rata-rata waktu pengiriman meningkat menjadi 45,2 detik. Meskipun demikian, kedua hasil



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

tersebut masih berada di bawah ambang waktu 60 detik yang ditetapkan pada kebutuhan non-fungsional (NF.1.3).

Temuan ini menunjukkan bahwa mekanisme *background scheduler* dan arsitektur *Publish/Subscribe* (Pub/Sub) mampu mempertahankan pengiriman notifikasi secara andal meskipun terjadi penurunan kualitas jaringan. Hasil tersebut sejalan dengan Siam et al. (2025), yang menyatakan bahwa peringatan kegagalan sistem sebaiknya diterima oleh tim SOC dalam waktu kurang dari satu menit agar respons mitigasi dapat dilakukan lebih cepat.

Hasil pengujian non-fungsional pada aspek keamanan data (*Security*) yang mencakup skenario NF.2.1 hingga NF.2.3 pada subbab sebelumnya menunjukkan bahwa arsitektur sistem telah berhasil mengimplementasikan prinsip *Security by Design* secara komprehensif. Pada tingkat *server*, sistem terbukti mencegah penyimpanan data sensitif dalam bentuk teks terang (*plaintext*). Seluruh kredensial infrastruktur (seperti kata sandi komponen Wazuh dan *API Key*) diamankan ke dalam format *ciphertext* acak menggunakan enkripsi simetris AES (NF.2.1), sedangkan kata sandi pengguna dilindungi menggunakan algoritma hashing (NF.2.2).

Pendekatan ini secara efektif memitigasi risiko eskalasi hak akses seandainya terjadi peretasan pada basis data. Lebih lanjut, pada tingkat interaksi pengguna, sistem memastikan bahwa *cookie* sesi autentikasi dilindungi secara ketat oleh atribut *HttpOnly* dan *Secure* (NF.2.3). Hal ini membuktikan bahwa aplikasi memiliki pertahanan yang kokoh terhadap upaya penyadapan sesi melalui serangan skrip sisi klien (*Cross-Site Scripting* / XSS).

Hasil pengujian non-fungsional pada aspek keandalan sistem (*Reliability & Fault Tolerance*) yang meliputi skenario NF.3.1 hingga NF.3.3 pada subbab sebelumnya menunjukkan kemampuan sistem yang sangat baik dalam mengisolasi kegagalan (*fault isolation*) agar tidak memicu kerusakan sistemik. Pada pengujian toleransi *node* (NF.3.1), terputusnya koneksi pada salah satu *server* Wazuh tidak menyebabkan antarmuka *dashboard*



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

mengalami *crash*, melainkan hanya memberikan indikator visual *Offline* secara spesifik pada komponen yang bermasalah.

Kemampuan sistem dalam mempertahankan operasional juga terlihat ketika menghadapi gangguan pada layanan pihak ketiga. Pada skenario kegagalan autentikasi API AlienVault OTX (NF.3.2) maupun kegagalan komunikasi dengan bot Telegram (NF.3.3), aplikasi web dan *background scheduler* tetap berjalan tanpa mengganggu fungsi utama sistem. Alih-alih berhenti atau menampilkan kesalahan yang membingungkan, sistem menangani kondisi tersebut melalui mekanisme *graceful error handling*, yaitu dengan menampilkan *toast notification* yang informatif kepada pengguna sekaligus mencatat detail kesalahan pada *backend log* untuk keperluan diagnosis.

Selain itu, selama pengujian tidak ditemukan kondisi yang menyebabkan informasi teknis internal, seperti *stack trace* atau *HTTP Error 500*, ditampilkan kepada pengguna. Temuan ini menunjukkan bahwa mekanisme penanganan kesalahan telah mampu mengisolasi gangguan pada layanan eksternal agar tidak memengaruhi pengalaman pengguna maupun operasional sistem secara keseluruhan. Berdasarkan ruang lingkup pengujian yang dilakukan, hasil tersebut mengindikasikan bahwa arsitektur Hub Manajemen Terpusat memiliki tingkat ketahanan yang baik terhadap gangguan layanan eksternal.

C. Analisis dan Evaluasi Efisiensi Operasional

Untuk mengukur dampak nyata dari implementasi Hub Manajemen Terpusat terhadap produktivitas kerja *Security Operations Center* (SOC), data empiris berupa durasi waktu dan jumlah langkah (*step count*) dikuantifikasi ke dalam bentuk persentase efisiensi. Mengacu pada adaptasi metrik *Deployment Time Reduction Rate* (DTRR), perhitungan persentase reduksi beban kerja dilakukan secara individual pada ke-10 aksi operasional dengan membandingkan data waktu hasil sistem manual dikurangi sistem usulan. Hasil dari komparasi tersebut dirangkum pada Tabel 4.44 berikut:



Tabel 4.44 Tingkat Efisiensi (Reduction Rate) per Aksi Operasional

ID Aksi	Deskripsi Skenario Operasional	Reduksi Waktu (%)	Reduksi Langkah (%)
Aksi 1	Inventarisasi Komponen Baru	41,27%	50,00%
Aksi 2	Pemeliharaan Kredensial	66,22%	25,00%
Aksi 3	Pemantauan Kesehatan Multi-Komponen	65,52%	60,00%
Aksi 4	Inventarisasi Daftar Agen (<i>Agent List</i>)	71,43%	80,00%
Aksi 5	Investigasi <i>On-Demand</i> (OTX)	32,00%	33,33%
Aksi 6	Investigasi <i>Alert</i> (<i>Click-to-Investigate</i>)	47,37%	25,00%
Aksi 7	Konfigurasi API <i>Threat Intelligence</i>	64,10%	50,00%
Aksi 8	Konfigurasi Notifikasi (Telegram)	71,21%	60,00%
Aksi 9	Uji Coba Integrasi (<i>Connectivity Test</i>)	43,18%	80,00%
Aksi 10	Identifikasi Akar Masalah Insiden <i>Down</i>	61,19%	75,00%

Meskipun Tabel 4.44 menunjukkan variasi tingkat efisiensi pada setiap tugas, penilaian terhadap kinerja sistem perlu dilakukan secara menyeluruh, bukan berdasarkan masing-masing aktivitas secara terpisah. Oleh karena itu, untuk memperoleh gambaran efisiensi sistem secara keseluruhan, dilakukan perhitungan *Average Reduction Rate* berdasarkan selisih antara rata-rata kumulatif waktu pada prosedur eksisting (*baseline*) dan rata-rata kumulatif waktu pada sistem usulan.

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Evaluasi terhadap penghematan waktu operasional (*Time on Task*) dirumuskan sebagai berikut:

$$EfisiensiWaktu = \frac{\text{Rata-rata Waktu}_{Manual} - \text{Rata-rata Waktu}_{Usulan}}{\text{Rata-rata Waktu}_{Manual}} \times 100\%$$

$$EfisiensiWaktu = \frac{51,6 \text{ detik} - 21,5 \text{ detik}}{51,6 \text{ detik}} \times 100\%$$

$$EfisiensiWaktu = \frac{30,1}{51,6} \times 100\% = 58,33\%$$

Selanjutnya, evaluasi terhadap penurunan beban interaksi pengguna (*Step Count*) dirumuskan sebagai berikut:

$$EfisiensiLangkah = \frac{\text{Rata-rata Langkah}_{Manual} - \text{Rata-rata Langkah}_{Usulan}}{\text{Rata-rata Langkah}_{Manual}} \times 100\%$$

$$EfisiensiLangkah = \frac{5,0 \text{ langkah} - 2,4 \text{ langkah}}{5,0 \text{ langkah}} \times 100\%$$

$$EfisiensiLangkah = \frac{2,6}{5,0} \times 100\% = 52,00\%$$

Guna memperjelas perbedaan performa antara metode konvensional dan sistem usulan, hasil perhitungan matematis tersebut selanjutnya divisualisasikan ke dalam bentuk visual. Penyajian ini memudahkan pembaca dalam membandingkan durasi eksekusi pada setiap aktivitas tanpa harus menafsirkan nilai numerik secara terpisah. Gambar 4.86 mengilustrasikan perbandingan waktu eksekusi kedua metode dalam satuan detik, sehingga perbedaan efisiensi dapat diamati dengan lebih jelas.

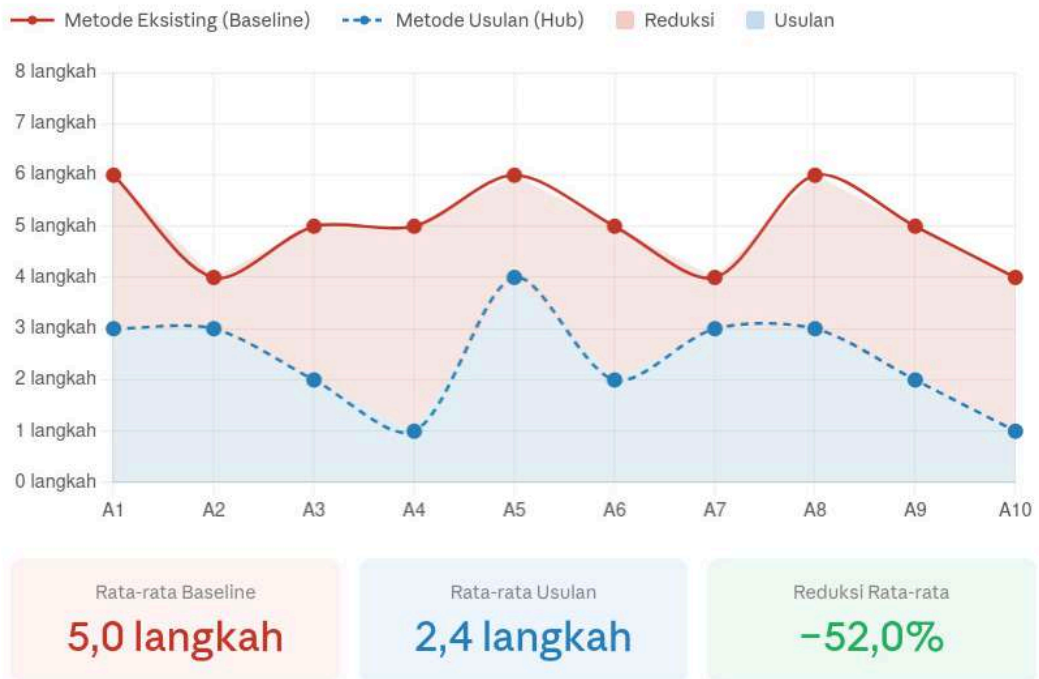


Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

PERBANDINGAN LANGKAH

Jumlah Langkah per Aksi Operasional



Gambar 4.86 Grafik Komparasi Langkah per Aksi Operasional Eksisting & Usulan

Hasil dari Gambar 4.86 memperlihatkan bahwa seluruh aktivitas operasional mengalami penurunan jumlah langkah setelah menggunakan sistem usulan. Pada metode eksisting, sebagian besar aktivitas memerlukan 4 hingga 6 langkah untuk diselesaikan, sedangkan pada sistem usulan kebutuhan interaksi berkurang menjadi 1 hingga 4 langkah. Penurunan ini menunjukkan bahwa beberapa tahapan manual berhasil diintegrasikan dan diotomatisasi ke dalam satu antarmuka terpadu.

Secara keseluruhan, rata-rata jumlah langkah berkurang dari 5,0 langkah pada metode eksisting menjadi 2,4 langkah pada sistem usulan, atau setara dengan reduksi sebesar 52,0%. Hasil ini mengindikasikan bahwa sistem tidak hanya mempercepat penyelesaian aktivitas, tetapi juga menyederhanakan alur kerja administrator melalui pengurangan klik, perpindahan halaman, dan interaksi yang bersifat repetitif.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Pada analisis komparasi selanjutnya Gambar 4.87 memetakan perbandingan jumlah interaksi, seperti klik dan navigasi, yang harus dilakukan analis pada setiap aktivitas operasional. Visualisasi ini memberikan gambaran mengenai besarnya beban interaksi yang dapat dikurangi melalui penerapan sistem usulan dibandingkan dengan pendekatan konvensional.



Gambar 4.87 Grafik Komparasi detik per Operasional Eksisting & Usulan

Hasil visualisasi pada Gambar 4.87 menunjukkan bahwa sistem usulan secara konsisten mampu mengurangi waktu penyelesaian pada seluruh aktivitas operasional dibandingkan metode eksisting. Rata-rata durasi eksekusi berkurang dari 51,6 detik menjadi 21,5 detik, atau setara dengan reduksi sebesar 58,3%.

Penurunan ini mengindikasikan bahwa integrasi berbagai proses administrasi ke dalam satu antarmuka terpadu mampu mengurangi waktu yang dihabiskan untuk perpindahan antar aplikasi, autentikasi berulang, dan eksekusi langkah-langkah manual, sehingga aktivitas operasional dapat diselesaikan dengan lebih efisien.

**Hak Cipta :**

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Kedua bukti efisiensi operasional baik langkah dan reduksi waktu menjelaskan seberapa efektif sistem usulan menjawab permasalahan operasional yang dijabarkan pada Bab I. Secara arsitektural, keberhasilan optimalisasi tidak hanya berhasil mempercepat proses klik, namun juga mengubah tiga fondasi utama cara analisis SOC bekerja:

1. Pengurangan Beban Kognitif dan *Context Switching*

Salah satu sumber beban kerja pada operasional SOC berasal dari banyaknya perpindahan antar aplikasi (*context switching*) serta aktivitas administratif yang harus dilakukan untuk memperoleh informasi yang dibutuhkan. Oleh karena itu, sistem usulan dirancang untuk mengintegrasikan berbagai proses administrasi ke dalam satu antarmuka sehingga kebutuhan berpindah aplikasi dan melakukan langkah-langkah berulang dapat dikurangi.

Dampak implementasi tersebut terlihat pada hasil pengujian Aksi 4 (Inventarisasi Daftar Agen). Pada metode eksisting, administrator harus melalui lima langkah, mulai dari membuka terminal, melakukan autentikasi API untuk memperoleh *access token*, menggunakan token tersebut untuk mengambil data, hingga meninjau keluaran JSON secara manual. Pada sistem usulan, proses tersebut disederhanakan menjadi satu langkah, yaitu membuka halaman Wazuh Agents, sehingga jumlah interaksi berkurang sebesar 80%. Penyederhanaan ini dimungkinkan karena sistem secara otomatis melakukan autentikasi, penarikan data dari beberapa Wazuh Manager secara paralel, serta menyajikan hasil agregasi dalam bentuk yang siap digunakan.

Dengan berkurangnya aktivitas administratif tersebut, pengguna dapat memperoleh informasi yang diperlukan tanpa lagi berinteraksi langsung dengan *command line*, *token API*, maupun struktur data mentah. Kondisi ini membantu mengurangi beban kognitif selama proses operasional sehingga waktu dan perhatian pengguna dapat lebih difokuskan pada analisis hasil pemantauan dan pengambilan keputusan berdasarkan informasi yang telah tersedia.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

2. Perubahan Mekanisme Pemantauan dari Reaktif (*Pull*) menjadi Proaktif (*Push*)

Pada pendekatan pemantauan tradisional, kondisi infrastruktur umumnya diketahui melalui mekanisme *pull*, yaitu administrator atau analis harus secara berkala memeriksa setiap komponen untuk memastikan layanan masih beroperasi dengan normal. Pendekatan ini bergantung pada pemeriksaan manual sehingga terdapat kemungkinan gangguan tidak segera diketahui apabila proses pengecekan terlambat atau terlewat. Kondisi tersebut berpotensi menimbulkan *blind spot* operasional, terutama ketika suatu layanan berhenti beroperasi tanpa menghasilkan indikasi yang langsung terlihat oleh tim SOC.

Hasil pengujian pada Aksi 10 (Identifikasi Insiden *Down*) menunjukkan bahwa sistem usulan mengubah mekanisme tersebut dengan menerapkan pendekatan *push*. Waktu yang diperlukan untuk mengetahui adanya gangguan berkurang dengan efisiensi sebesar 61,19% karena sistem tidak lagi bergantung pada pemeriksaan manual melalui terminal. Sebaliknya, *background scheduler* (Sentinel) secara berkala memantau kondisi setiap layanan dan memanfaatkan *Redis State Machine* untuk mendeteksi perubahan status. Ketika sebuah komponen tidak lagi merespons, sistem secara otomatis mengirimkan notifikasi ‘*CRITICAL HEALTH ALERT*’ ke Telegram beserta informasi mengenai komponen yang terdampak. Dengan mekanisme ini, administrator dapat memperoleh informasi gangguan lebih cepat tanpa harus melakukan pengecekan manual, sehingga visibilitas terhadap kondisi infrastruktur meningkat dan risiko *silent failure* dapat dikurangi pada ruang lingkup sistem yang diuji.

3. Konsolidasi Konfigurasi melalui *Single Source of Truth* (SSOT)

Pengelolaan infrastruktur Wazuh yang terdiri atas beberapa *node* menuntut agar konfigurasi tetap konsisten pada seluruh komponen tanpa mengganggu operasional sistem. Tantangan tersebut tercermin pada Aksi 1, 2, 4, dan 8 yang berkaitan dengan penambahan komponen, pembaruan kredensial,



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

inventarisasi agen, serta konfigurasi notifikasi. Di antara seluruh aktivitas tersebut, Aksi 8 (Konfigurasi Notifikasi) mencatat reduksi waktu terbesar, yaitu sebesar 71,21%. Hasil ini menunjukkan bahwa proses konfigurasi yang sebelumnya dilakukan melalui serangkaian perintah pada tingkat sistem operasi dapat disederhanakan menjadi konfigurasi berbasis antarmuka web.

Selain mempercepat proses administrasi, pendekatan ini juga menyederhanakan pengelolaan konfigurasi melalui mekanisme *Single Source of Truth* (SSOT). Seluruh perubahan konfigurasi dilakukan dari satu antarmuka, kemudian disimpan secara terpusat di PostgreSQL dengan kredensial yang dienkripsi menggunakan AES. Informasi konfigurasi yang telah diperbarui selanjutnya diterapkan melalui mekanisme *hot reload* dan didukung oleh Redis untuk memperbarui data operasional tanpa memerlukan perubahan konfigurasi secara manual pada setiap layanan. Dengan mekanisme tersebut, proses pembaruan konfigurasi dapat dilakukan dengan lebih sederhana sekaligus mengurangi potensi gangguan terhadap operasional pemantauan selama proses perubahan.

Secara keseluruhan, hasil analisis menunjukkan bahwa Hub Manajemen Terpusat mampu menyederhanakan berbagai aktivitas operasional SOC melalui integrasi pengelolaan infrastruktur, pengayaan *Threat Intelligence*, dan notifikasi status sistem ke dalam satu antarmuka. Berdasarkan ruang lingkup pengujian yang dilakukan, pendekatan ini berhasil mengurangi jumlah langkah kerja, mempercepat waktu penyelesaian berbagai aktivitas operasional, serta mengurangi kebutuhan perpindahan antar aplikasi dalam proses administrasi dan investigasi.



BAB V PENUTUP

5.1. Simpulan

Penelitian ini menghasilkan sebuah Hub Manajemen Terpusat yang mengintegrasikan pengelolaan infrastruktur multi-Wazuh, pengayaan *Threat Intelligence*, dan notifikasi status sistem ke dalam satu platform. Untuk mendukung pengelolaan data, sistem memanfaatkan PostgreSQL sebagai pusat penyimpanan konfigurasi (*Single Source of Truth*) dengan mekanisme enkripsi AES untuk melindungi kredensial, sedangkan Redis digunakan sebagai penyimpanan sementara untuk mendukung proses *caching* dan pemantauan status sistem secara *real-time*. Berdasarkan hasil pengujian yang dilakukan, seluruh skenario fungsional yang dirancang berhasil dijalankan sesuai dengan keluaran yang diharapkan. Selain itu, pengujian non-fungsional menunjukkan bahwa sistem mampu mempertahankan waktu respons, menangani gangguan layanan eksternal melalui mekanisme yang sesuai, serta menjaga keamanan data sensitif selama proses operasional. Integrasi dengan AlienVault OTX juga memungkinkan proses pencarian *Indicator of Compromise* (IoC) dilakukan secara asinkron, sementara modul Sentinel secara otomatis memantau kondisi infrastruktur dan mengirimkan notifikasi ke Telegram ketika terdeteksi perubahan status layanan.

Menjawab tujuan utamanya, sistem yang diusulkan mampu menyederhanakan proses kerja sekaligus mengurangi waktu yang diperlukan untuk menyelesaikan berbagai aktivitas administrasi dan pemantauan. Hasil pengujian menunjukkan rata-rata durasi penyelesaian tugas berkurang sebesar 58,33%, sedangkan jumlah langkah interaksi pengguna berkurang sebesar 52,0%. Pengurangan tersebut dicapai melalui penyatuan berbagai fungsi ke dalam satu antarmuka sehingga kebutuhan berpindah aplikasi, melakukan autentikasi berulang, serta menyalin data secara manual dapat diminimalkan. Dengan demikian, Hub Manajemen Terpusat yang dikembangkan pada penelitian ini memberikan pendekatan yang lebih sederhana, terintegrasi, dan proaktif dalam mendukung operasional *Security Operations Center* (SOC), sehingga administrator dan analis dapat memperoleh informasi yang dibutuhkan dengan



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

lebih cepat untuk mendukung proses pengambilan keputusan selama penanganan insiden.

5.2. Saran

Penelitian ini menghasilkan sebuah arsitektur yang dapat menjadi fondasi bagi pengembangan sistem pada penelitian selanjutnya. Meskipun fungsi utama yang dirancang telah diimplementasikan dan dievaluasi, masih terdapat sejumlah peluang untuk memperluas cakupan sistem sesuai kebutuhan operasional *Security Operations Center* (SOC). Salah satunya adalah memperluas mekanisme pemantauan otomatis yang saat ini diterapkan pada komponen Wazuh agar juga mencakup perangkat infrastruktur lain, seperti *firewall*, *load balancer*, maupun layanan jaringan yang bersifat kritis. Selain itu, pola integrasi API asinkron yang digunakan pada sistem dirancang secara modular sehingga memungkinkan penambahan sumber *Threat Intelligence* lain, seperti VirusTotal, MISP, atau IBM X-Force, tanpa memerlukan perubahan yang signifikan pada arsitektur yang telah dibangun. Integrasi beberapa sumber tersebut diharapkan dapat memperkaya informasi pendukung yang tersedia selama proses investigasi insiden.

Pengembangan berikutnya juga dapat difokuskan pada peningkatan mekanisme notifikasi. Pada penelitian ini, notifikasi Telegram digunakan untuk menyampaikan perubahan status kesehatan infrastruktur secara *real-time*. Ke depan, mekanisme yang sama dapat diperluas untuk mendistribusikan peringatan ancaman siber yang berasal dari hasil analisis log maupun *Threat Intelligence*. Apabila volume peringatan meningkat, penerapan algoritma *Machine Learning* dapat dipertimbangkan untuk membantu mengelompokkan, memprioritaskan, dan menyaring notifikasi berdasarkan tingkat risiko. Pendekatan ini berpotensi mengurangi jumlah peringatan yang kurang relevan (*alert fatigue*), sehingga analis dapat lebih memfokuskan perhatian pada insiden yang memiliki prioritas penanganan lebih tinggi.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

DAFTAR RIWAYAT HIDUP PENULIS



Christian Nataniel Yosua Purba menempuh pendidikan menengah di SMA Swasta Methodist dengan peminatan Ilmu Pengetahuan Alam (IPA). Setelah menyelesaikan pendidikan tersebut, penulis melanjutkan studi pada Program Studi Sarjana Terapan Teknik Multimedia dan Jaringan, Jurusan Teknik Informatika dan Komputer di Politeknik Negeri Jakarta (PNJ). Selama masa perkuliahan, penulis aktif mengembangkan kompetensi akademik

maupun non-akademik melalui berbagai organisasi, kepanitiaan, dan kegiatan pengembangan diri. Selama perkuliahan penulis menjadi bagian 2 organisasi yakni KSM Computer Student Club (CSC) divisi Cyber Security serta menjadi Mentor divisi Speech di Polytechnic English Club (PEC). Selain itu, penulis berkesempatan menjadi panitia pada ajang nasional Kompetisi Mahasiswa Bidang Informatika Politeknik (KMIPN VI) dan konferensi International Conference on Computing, Communication, and Information Engineering (IC2IE), serta mengikuti Program Studi Independen Bersertifikat Kampus Merdeka di bidang Digital Forensik. Selain berbagai pengalaman tersebut, penulis meraih Juara I Nasional pada kompetisi Dashboard Analysis Competition oleh GELAR RASA UPNV Jatim.



DAFTAR PUSTAKA

- Aare, C.R. 2025. "Scalable SIEM Architectures for Global Enterprises: Engineering Real-Time Visibility with Splunk". *Zenodo (CERN European Organization for Nuclear Research)*, 4(8). <https://doi.org/10.5281/zenodo.16786607>.
- Abiade, O. 2024. *Enhancing Cybersecurity Through Predictive Threat Intelligence and Automated Incident Handling*. EasyChair Preprint 14890. <https://easychair.org/publications/preprint/FqhD>.
- Adrian, A.Z.A., Megantara, R.A. dan Al Zami, F. 2026. "Hybrid Multilayer Architecture Integrating Suricata, Wazuh, and Cyber Threat Intelligence for Drive-by-Download Malvertising Detection". *Sinkron*, 10(1), 161–168. <https://doi.org/10.33395/sinkron.v10i1.15616>.
- Allegretta, M., Siracusano, G., Gonzalez, R. dan Gramaglia, M. 2023. "Are Crowd-Sourced CTI Datasets Ready for Supporting Anti-Cybercrime Intelligence?". *Computer Networks*, 234, 109920. <https://doi.org/10.1016/j.comnet.2023.109920>.
- Amami, R., Charfeddine, M. dan Masmoudi, S. 2024. "Exploration of Open Source SIEM Tools and Deployment of an Appropriate Wazuh-Based Solution for Strengthening Cyberdefense". *2024 10th International Conference on Control, Decision and Information Technologies (CoDIT)*, 8, 1–7. <https://doi.org/10.1109/codit62066.2024.10708476>.
- Andali, D.A.S. dan Susetyo, Y.A. 2025. "Performance Analysis of NoSQL Databases: MongoDB Document Store and Redis Key-Value Store in Microservices-Based Applications Using Flask". *International Journal Software Engineering and Computer Science (IJSECS)*, 5(2), 691–705. <https://doi.org/10.35870/ijsecs.v5i2.4206>.
- Arif, I., Tariq, A. dan Barchuk, B. 2025. "The Cost of Fragmentation: Measuring Time, Spend and Risk in Personal Cybersecurity Tool Stacks". *World Journal of Advanced Engineering Technology and Sciences*, 16(3), 334–363. <https://doi.org/10.30574/wjaets.2025.16.3.1350>.
- AwanPintar.id. 2025. *Laporan Ancaman Digital di Indonesia Semester 1 Tahun 2025*. PT PROSPERITA SISTEM INDONESIA. <https://www.awanpintar.id/publikasi/>.

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengemukakan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

- Badan Siber dan Sandi Negara (BSSN). 2024. *Lanskap Keamanan Siber Indonesia 2024*. Jakarta Selatan: Id-SIRTII /CC – BSSN. <https://www.bssn.go.id/monitoring-keamanan-siber/>. [9 Januari 2026].
- Barman, U. dan Joshi, K. 2025. "Polyglot Persistence - Usage and Challenges". *Journal of Artificial Intelligence, Machine Learning and Data Science*, 3(4), 3079–3089. <https://doi.org/10.51219/jaimld/utpal-barma/633>.
- Baruwal Chhetri, M., Tariq, S., Singh, R., Jalalvand, F., Paris, C. dan Nepal, S. 2024. "Towards Human-AI Teaming to Mitigate Alert Fatigue in Security Operations Centres". *ACM Transactions on Internet Technology*, 24(3). <https://doi.org/10.1145/3670009>.
- Bell, D. 2023. *Explore the UML Sequence Diagram*. IBM Developer. <https://developer.ibm.com/articles/the-sequence-diagram/>.
- Burke, G. 2025. *SIEM SERIES WAZUH*. Objectfirst.com. <https://objectfirst.com/blog/siem-series-wazuh/>.
- Chaitanya, A.K., Jignyasa, G., Keerthi, S. dan Reddy, G.H. 2025. "Fortifying Cyber Defenses: Empowering Web Application Firewalls Through Threat Intelligence". *International Journal of Information Technology and Computer Engineering*, 13(1), 242–249. <https://ijitce.org/index.php/ijitce/article/view/881>. [18 Januari 2026].
- CrowdStrike. 2023. *CrowdStrike 2023 Threat Hunting Report Reveals Identity-Based Attacks and Hands-on-Keyboard Activity on the Rise as Adversaries Look to Bypass Defenses*. Crowdstrike.com. <https://www.crowdstrike.com/en-us/press-releases/crowdstrike-releases-2023-threat-hunting-report/>.
- CrowdStrike. 2024. *2024 CrowdStrike Global Threat Report: From Breakout to Breach in Under Three Minutes; Cloud Infrastructure Under Attack*. Crowdstrike.com. <https://www.crowdstrike.com/en-us/press-releases/2024-crowdstrike-global-threat-report-release/>.
- CrowdStrike. 2025. *CrowdStrike Releases 2025 Global Threat Report: Cyber Threats Reach New Highs*. Crowdstrike.com. <https://www.crowdstrike.com/en-us/press-releases/crowdstrike-releases-2025-global-threat-report/>.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

- Dharma, B.S. dan Muslikh, A.R. 2025. "Implementasi Rate Limiting dan Bot Telegram untuk Mitigasi Serangan HTTP GET Flood". *Journal of Information System and Application Development*, 3(1). <https://doi.org/10.26905/jisad.v3i1.15398>.
- Domo, Inc. 2026. *What Is Real-Time Data? Definition, Benefits, and Best Practices for 2026*. [www.domo.com](https://www.domo.com/glossary/what-is-real-time-data). <https://www.domo.com/glossary/what-is-real-time-data>.
- Gault, J. 2025. *Understanding Real-Time Analytics*. dbt Labs. <https://www.getdbt.com/discover/understanding-real-time-analytics>.
- Grobler, M., Gaire, R. dan Nepal, S. 2021. "User, Usage and Usability: Redefining Human Centric Cyber Security". *Frontiers in Big Data*, 4. <https://doi.org/10.3389/fdata.2021.583723>.
- Hardika, B., Kurniawan, M.D., Adzka, M., Prastowiyono, D., Banyubasa, A., Wicaksono, A. dan Nasir, M. 2024. "Pengujian Blackbox Testing Website Garuda Farm Menggunakan Teknik Equivalence Partitioning". *Jurnal Kridatama Sains dan Teknologi*, 6(2), 740–753. <https://doi.org/10.53863/kst.v6i02.1420>.
- Hegde, R.C. 2020. "Low Latency Message Brokers". *International Research Journal of Engineering and Technology (IRJET)*. <https://www.semanticscholar.org/paper/Low-Latency-Message-Brokers-Hegde/522190ca1b32b734d98ac731feacfe6ff2763f9c>.
- Hugo, V. dan Proust, M. 2022. "Integrating Firewalls with SIEM and SOAR Platforms for Automated Threat Response". *International Journal of Trend in Scientific Research and Development (IJTSRD)*, 6(3), 2315–2323. <https://www.ijtsrd.com/papers/ijtsrd49651.pdf>.
- Hyun, G., Oak, J., Kim, D. dan Kim, K. 2024. "The Impact of an Automation System Built with Jenkins on the Efficiency of Container-Based System Deployment". *Sensors*, 24(18), 6002. <https://doi.org/10.3390/s24186002>.
- IBM Security dan Ponemon Institute. 2020. *IBM Study: Security Response Planning on the Rise, But Containing Attacks Remains an Issue*. IBM News Room. <https://newsroom.ibm.com/2020-06-30-IBM-Study-Security-Response-Planning-on-the-Rise-But-Containing-Attacks-Remains-an-Issue>. [17 Juli 2021].



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

- Islam, M.R. dan Rafique, R. 2024. "Wazuh SIEM for Cyber Security and Threat Mitigation in Apparel Industries". *International Journal of Engineering Materials and Manufacture*, 9(4), 136–144. <https://doi.org/10.26776/ijemm.09.04.2024.02>.
- Ismail, Kurnia, R., Brata, Z.A., Nelistiani, G.A., Heo, S., Kim, H. dan Kim, H. 2025. "Toward Robust Security Orchestration and Automated Response in Security Operations Centers with a Hyper-Automation Approach Using Agentic Artificial Intelligence". *Information*, 16(5), 365. <https://doi.org/10.3390/info16050365>.
- ISO. 2018. *Ergonomics of Human-System Interaction — Part 11: Usability: Definitions and Concepts*. Iso.org. <https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-2:v1:en>.
- Izzatillah, T.I., Fania, F. dan Zulfata, Z. 2024. "The Effectiveness of Snort in DDoS Attack Scenarios". *IC-ITECHS*, 5(1), 603–612. <https://doi.org/10.32664/ic-itechs.v5i1.1588>.
- Jovanović, M. 2024. *Caching in ASP.NET Core: Improving Application Performance*. Milanjovanovic.tech. <https://www.milanjovanovic.tech/blog/caching-in-aspnetcore-improving-application-performance>. [26 Maret 2026].
- Kaptsov, L. 2025. "Using Redis for Caching Optimization in High-Traffic Web Applications". *International Journal of Advanced Multidisciplinary Research and Studies*, 5(4), 1714–1722. <https://doi.org/10.62225/2583049x.2025.5.4.4839>.
- Kelton, C., Ryoo, J., Balasubramanian, A. dan Das, S. 2017. "Improving User Perceived Page Load Time Using Gaze". *Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI '17)*, Boston, MA, USA. <https://www.usenix.org/system/files/conference/nsdi17/nsdi17-kelton.pdf>.
- Kumar, K.S. 2025. "Next-Generation Security Operations: Leveraging Automation for Proactive Threat Mitigation". *International Journal of Scientific Research in Engineering and Management*, 9(4), 1–9. <https://doi.org/10.55041/ijrem43432>.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengemukakan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Kummarapurugu, C.S. 2024. "Architectural Framework for Threat Intelligence Integration with SIEM and SOAR in Hybrid Cloud Security Environments". *International Journal of Innovative Research and Creative Technology (IJIRCT)*, 10(2), 1–9. <https://www.ijirct.org/viewPaper.php?paperId=2411031>.

Kurniawan, A., Aufar, T.Q. dan Agustin, M. 2024. "Configuration Management dengan Ansible dan Telegram untuk Automasi Laboratorium Komputer". *Multinetics*, 10(1), 50–59. <https://doi.org/10.32722/multinetics.v10i1.6360>.

Lucidchart. 2018. *The Basics of Documenting and Analyzing Your As-Is Process*. www.lucidchart.com. <https://www.lucidchart.com/blog/as-is-process-analysis>.

Matin, M., Fadhilahman, Kurniawan, A., Zain, A.R. dan Agustin, M. 2025. "Sistem Monitoring Keamanan pada Data Center Berbasis Security Information and Event Management (SIEM) dengan Wazuh dan IDS". *Journal of Innovative and Creativity*, 5(1), 530–539. <https://doi.org/10.31004/joecy.v5i1.1761>.

Muhammad, R., Ismail, S.A. dan Hassan, N.H. 2024. "Botnet Detection and Incident Response in Security Operation Center (SOC): A Proposed Framework". *International Journal of Advanced Computer Science and Applications*, 15(3). <https://doi.org/10.14569/ijacsa.2024.0150389>.

Nagar, G. 2018. "The Evolution of Security Operations Centers (SOCs): Shifting from Reactive to Proactive Cybersecurity Strategies". *International Journal of Scientific Research and Management (IJSRM)*, 6(9), 100–115. <https://doi.org/10.18535/ijssrm/v6i9.ec03>.

Neumann, F. dan Weber, C. 2021. "Automated Security Operations: Scaling Threat Response with SOAR and AI-Driven Playbooks". *International Journal of Trend in Scientific Research and Development (IJTSRD)*, 5(2), 1317–1323. www.ijtsrd.com/papers/ijtsrd38541.pdf.

Nova, F., Pratama, M.D. dan Prayama, D. 2022. "Wazuh sebagai Log Event Management dan Deteksi Celah Keamanan pada Server dari Serangan DoS". *JITSI: Jurnal Ilmiah Teknologi Sistem Informasi*, 3(1), 1–7. <https://doi.org/10.62527/jitsi.3.1.59>.

Object Management Group. 2017. *An OMG Unified Modeling Language Publication (OMG UML)*. <https://www.omg.org/spec/UML/2.5.1/PDF>.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Ofoegbu, K.D.O., Osundare, O.S., Ike, C.S., Fakeyede, O.G. dan Ige, A.B. 2024. "Proactive Cyber Threat Mitigation: Integrating Data-Driven Insights with User-Centric Security Protocols". *Computer Science & IT Research Journal*, 5(8), 2083–2106. <https://doi.org/10.51594/csitrj.v5i8.1493>.

Pressman, R.S. dan Maxim, B.R. 2014. *Software Engineering: A Practitioner's Approach*. New York: Mcgraw-Hill Education.

Pulyala, S.R., Desetty, A.G. dan Jangampet, V.D. 2019. "The Impact of Security Orchestration, Automation, and Response (SOAR) on Security Operations Center (SOC) Efficiency: A Comprehensive Analysis". *Turkish Journal of Computer and Mathematics Education*, 10(3), 1545–1549. <https://doi.org/10.61841/turcomat.v10i3.14323>.

Radview. 2026. *Benchmark Testing: How to Measure & Compare Performance*. Radview. <https://www.radview.com/blog/benchmark-testing/>. [26 Mei 2026].

Salunke, S.V. dan Ouda, A. 2024. "A Performance Benchmark for the PostgreSQL and MySQL Databases". *Future Internet*, 16(10), 382. <https://doi.org/10.3390/fi16100382>.

Seidl, M., Scholz, M., Huemer, C. dan Kappel, G. 2015. *UML @ Classroom*. Undergraduate Topics in Computer Science. Cham: Springer International Publishing.

Shafiyah, A., Nama, G.F. dan Pradipta, R.A. 2024. "Implementasi Wazuh Menggunakan Metode PPDIIO di Sistem Keamanan Jaringan PSDKU Universitas Lampung Waykanan Sebagai Deteksi dan Respon Serangan Siber". *Jurnal Informatika dan Teknik Elektro Terapan*, 12(2). <https://doi.org/10.23960/jitet.v12i2.4074>.

Shavaf, M., Nair, M., Fayas, M., Rasheed, N. dan Babu, A. 2025. "Optimising Security Threats with Open-Source Threat Intelligence". *IJRTI*, 10(5), 679. <https://www.ijrti.org/papers/IJRTI2505081.pdf>. [18 Januari 2026].

Sheeraz, M., Paracha, M.A., Ulhaq, M., Durad, H. dan Mosavi, A. 2023. "Effective Security Monitoring Using Efficient SIEM Architecture". *Human-Centric Computing and Information Sciences*, 13(23). <https://doi.org/10.22967/HCIS.2023.13.023>.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

- Shevchenko, V. 2025. "Strategies for Database Performance Optimization in High-Load Systems: A Review of PostgreSQL and Redis Use Cases". *International Journal of Computer (IJC)*, 56(1), 279–289. <https://ijcjournal.org/InternationalJournalOfComputer/article/view/2476>.
- Siam, A.A., Alazab, M., Awajan, A., Hasan, M.R., Obeidat, A. dan Faruqui, N. 2025. "IP SafeGuard—An AI-Driven Malicious IP Detection Framework". *IEEE Access*, 13, 90249–90261. <https://doi.org/10.1109/access.2025.3569289>.
- Sijabat, D.R. dan Evo, S. 2023. "Perancangan Security Information and Event Management (SIEM) untuk Mendeteksi Insiden pada Situs Web". *J-Intech: Journal of Information and Technology*, 11(1), 10–17. <https://doi.org/10.32664/j-intech.v11i1.860>.
- Singh, B. 2015. "Ensuring Data Integrity and Availability with Robust Database Security Protocols". *International Journal of Research in Electronics and Computer Engineering (IJRECE)*, 3(1). https://www.researchgate.net/publication/394294551_Ensuring_Data_Integrity_and_Availability_with_Robust_Database_Security_Protocols.
- StreamNative. 2025. *Latency Numbers Every Data Streaming Engineer Should Know*. Streamnative.io. <https://streamnative.io/blog/latency-numbers-every-data-streaming-engineer-should-know>. [6 Februari 2026].
- Sundaramurthy, S.C., Bardas, A.G., Case, J., Ou, X., Wesch, M., McHugh, J. dan Rajagopalan, S.R. 2015. "A Human Capital Model for Mitigating Security Analyst Burnout". *Symposium on Usable Privacy and Security (SOUPS)*, 347–359.
- Sutanto, A. dan Rakhman, A. 2025. "Experimental Evaluation of Wazuh-Grafana Integration for Real-Time Cyber Threat Detection in Resource-Constrained Environments". *Journal of Applied Informatics and Computing*, 9(5), 2783–2790. <https://doi.org/10.30871/jaic.v9i5.10404>.
- Thakker, A., More, A. dan Kumar, K. 2025. "Automated Defense Against Application-Layer Attacks on Windows Systems Using Wazuh and Shuffle". *International Journal of Education Science Technology and Engineering (IJESTE)*, 8(1), 45–57. <https://doi.org/10.36079/lamintang.ijeste-0801.842>.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengemukakan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Velamoo, G. dan Samanta, A. 2024. *Adaptive Timeout Strategies for Microservice Application*. MIT PRIMES 2024. <https://math.mit.edu/research/highschool/primes/materials/2024/Velamoor-Samanta.pdf>.

Vielberth, M., Bohm, F., Fichtinger, I. dan Pernul, G. 2020. "Security Operations Center: A Systematic Study and Open Challenges". *IEEE Access*, 8, 227756–227779. <https://doi.org/10.1109/access.2020.3045514>.

Visual Paradigm. 2016. *How to Develop As-Is and To-Be Business Process?*. Visual-paradigm.com. <https://www.visual-paradigm.com/tutorials/as-is-to-be-business-process.jsp>.

Wazuh. n.d. *Architecture - Getting Started with Wazuh · Wazuh Documentation*. documentation.wazuh.com. <https://documentation.wazuh.com/current/getting-started/architecture.html>.

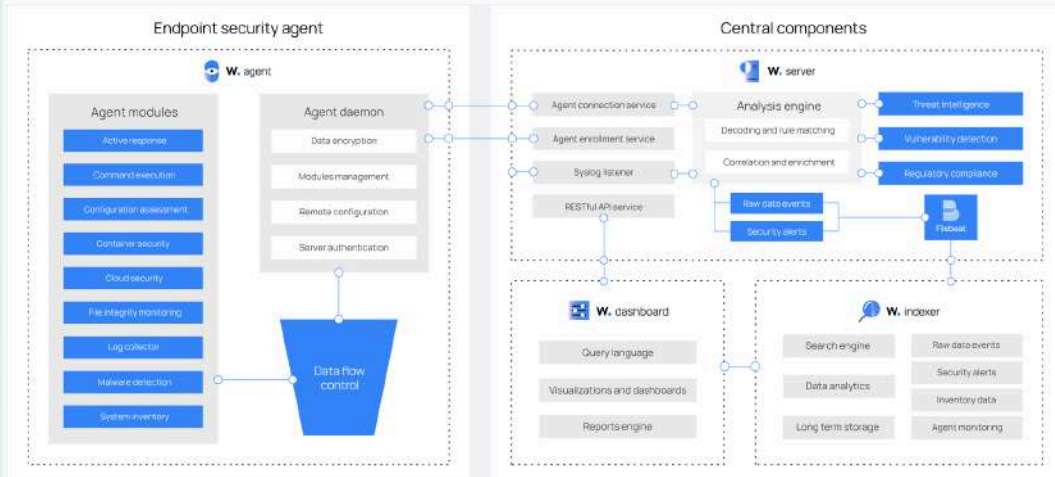
World Economic Forum. 2025. *Global Cybersecurity Outlook 2025*. https://reports.weforum.org/docs/WEF_Global_Cybersecurity_Outlook_2025.pdf#page=4.02.

**POLITEKNIK
NEGERI
JAKARTA**

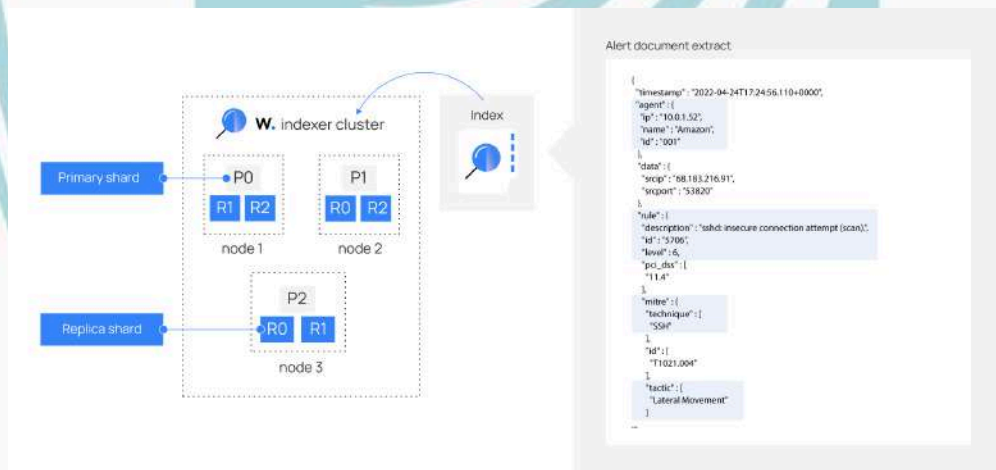


LAMPIRAN

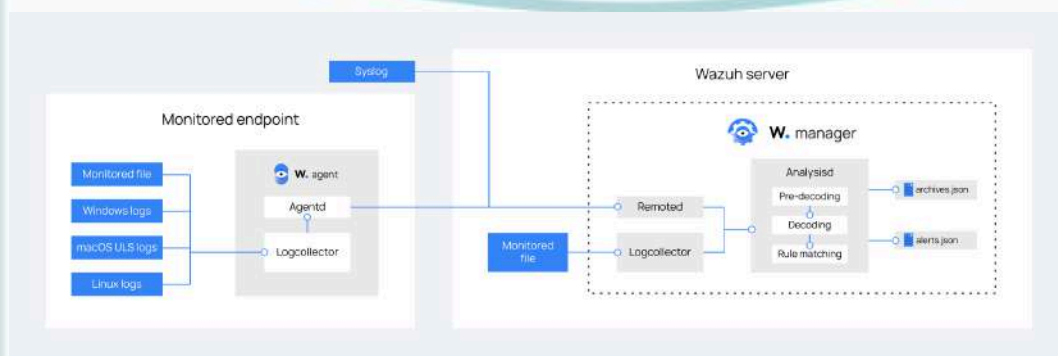
Lampiran 1 Diagram Arsitektur Wazuh



Lampiran 2 Diagram Arsitektur Komponen Wazuh Indexer



Lampiran 3 Diagram Arsitektur Komponen Wazuh Manager



Hak Cipta :

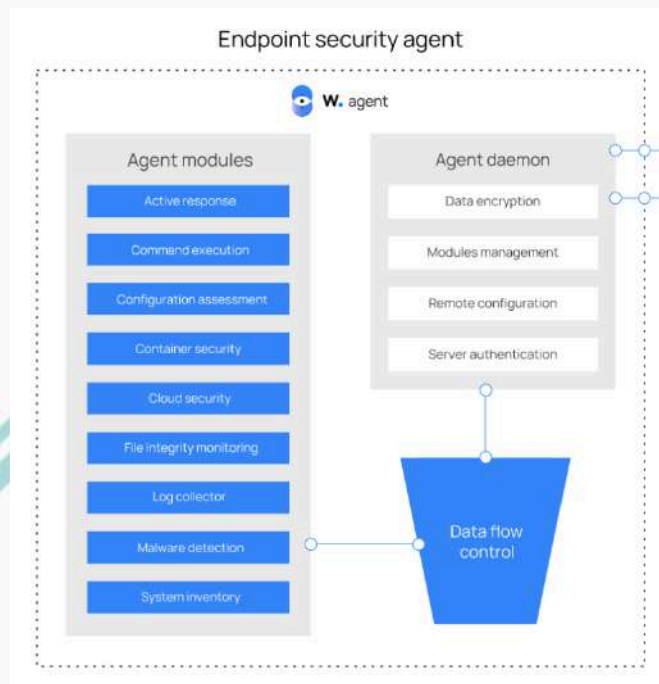
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengemukakan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Lampiran 4 Diagram Arsitektur Komponen Wazuh Agent



POLITEKNIK
NEGERI
JAKARTA



Lampiran 5 Tabel Hasil Pengujian dan Rujukan Bukti Waktu Muat Antarmuka (NF.1.1)

Kondisi Jaringan / Iterasi	Waktu Muat (s)	Screenshot Bukti
Kondisi Ideal (No Throttling)		
- Iterasi 1	2,27 s	The screenshot shows the Chrome DevTools Network tab with two requests: 'status?force=false' (0.8 kB, 185 ms) and 'status?force=false' (0.7 kB, 121 ms). The total load time is 2.27 s.
- Iterasi 2	2,96 s	The screenshot shows the Chrome DevTools Network tab with two requests: 'status?force=false' (0.8 kB, 171 ms) and 'status?force=false' (0.7 kB, 131 ms). The total load time is 2.96 s.

Hak Cipta :

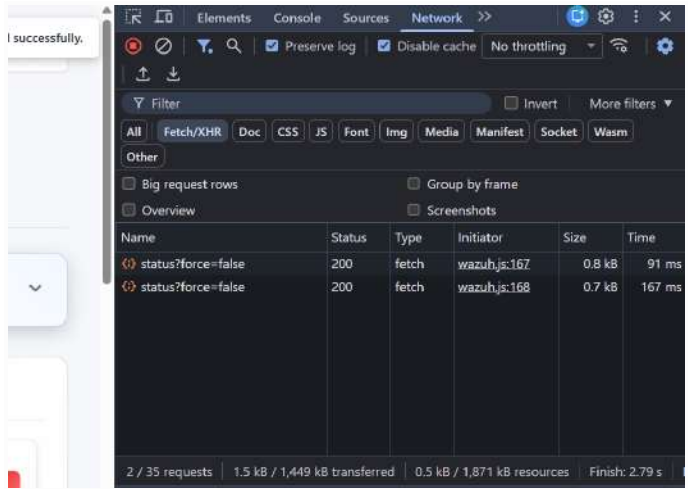
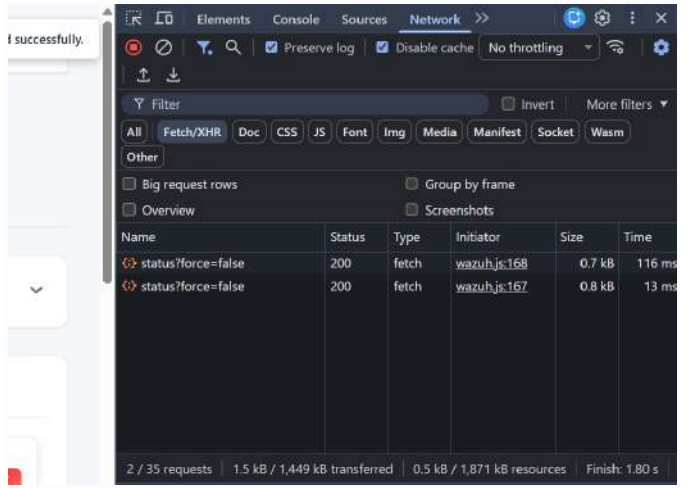
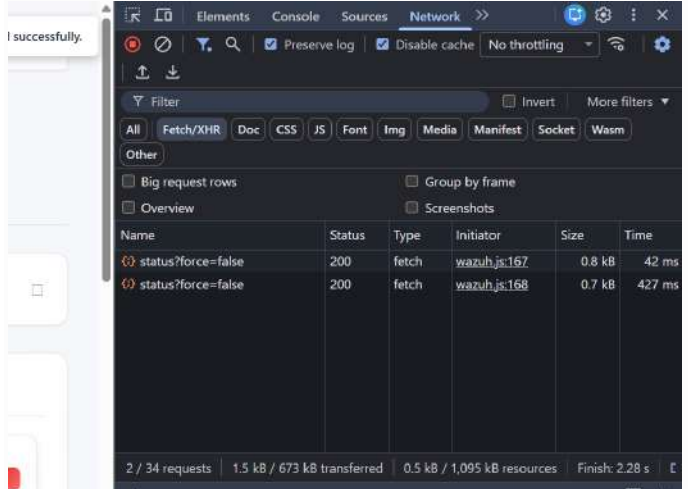
- Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
- Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang menggunakan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

(lanjutan)

- Iterasi 3	2,79 s	
- Iterasi 4	1,80 s	
- Iterasi 5	2,28 s	
Kondisi Moderat (<i>Fast 4G</i>)		

(lanjutan)



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

- Iterasi 1	6,23 s	1 successfully. 2 / 35 requests 1.5 kB / 1,449 kB transferred 0.5 kB / 1,871 kB resources Finish: 6.23 s
- Iterasi 2	9,78 s	1 successfully. 2 / 35 requests 1.5 kB / 1,449 kB transferred 0.5 kB / 1,871 kB resources Finish: 9.78 s
- Iterasi 3	4,87 s	successfully. 2 / 35 requests 1.5 kB / 1,449 kB transferred 0.5 kB / 1,871 kB resources Finish: 4.87 s



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

(lanjutan)

- Iterasi 4	7,01 s	
- Iterasi 5	3,54 s	
Kondisi Buruk (3G)		
- Iterasi 1	40,36 s	



(lanjutan)

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

- Iterasi 2	38,72 s	
- Iterasi 3	38,89 s	
- Iterasi 4	34,27 s	



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

(lanjutan)

- Iterasi 5	30,55 s	The screenshot shows the Network tab in Chrome DevTools. It displays two requests: 'status?force=false' with status 200 and type 'fetch'. The first request is initiated by 'wazuh.js:167' and has a size of 0.8 kB, taking 2.13 s. The second request is initiated by 'wazuh.js:168' and has a size of 0.7 kB, taking 2.16 s. The bottom status bar indicates '2 / 35 requests', '1.5 kB / 593 kB transferred', '0.5 kB / 1,016 kB resources', and a total finish time of '30.55 s'.
-------------	---------	--

POLITEKNIK
NEGERI
JAKARTA



Lampiran 6 Simulasi Gagal Fetch untuk Analyze Threat Intelligence (NF.1.2)

Threat Intelligence Search
Enter an Indicator of Compromise (e.g., IP, Domain, Hash) to analyze.
Data is provided by AlienVault OXZ.

Q 44d88612fea8a6f36de82e1276abb02f **ANALYZE**

Example indicators:
192.168.1.1 malicious-site.com 44d88612fea8a6f36de82e1276abb02f

Analysis Failed for: 44d88612fea8a6f36de82e1276abb02f
Failed to retrieve analysis result

Name	Status	Type	Initiator	Size	Time
alerts?page=1&per_page=1	(failed) ne...	fetch	api.js:149	0.0 kB	1 ms
alerts?page=1&per_page=1	(failed) ne...	fetch	api.js:149	0.0 kB	1 ms
alerts?page=1&per_page=1	(failed) ne...	fetch	api.js:149	0.0 kB	1 ms
analyze	(failed) ne...	fetch	intelligence.js:81	0.0 kB	6 ms
alerts?page=1&per_page=1	(failed) ne...	fetch	api.js:149	0.0 kB	1 ms
analyze	202	fetch	intelligence.js:81	0.6 kB	311 ms
alerts?page=1&per_page=1	200	fetch	api.js:149	2.1 kB	174 ms
status	(failed) ne...	fetch	api.js:182	0.0 kB	1 ms
status	(failed) ne...	fetch	api.js:183	0.0 kB	1 ms
d54b9244-b52c-459d-8a28-0224e0f5...	(failed) ne...	fetch	intelligence.js:113	0.0 kB	2 ms
alerts?page=1&per_page=1	(failed) ne...	fetch	api.js:149	0.0 kB	1 ms

83 / 176 requests 266 kB / 3,483 kB transferred 248 kB / 4,906 kB resources Finish 3.8 min DOMContentLoaded

Lampiran 7 Status Down Ketika Compose Wazuh Paused (NF.1.3.)

Configured Indexers

- Chris Indexer
https://192.168.111.6:9200
- Honeypot

```

RX errors 0 dropped 0 overruns 0 frame 0
TX packets 14113617 bytes 4749317261 (4.7 GB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

chris@lst-thread-intel-risk:~$ ls
lamis-docker ldap-auto opencti wazuh-agent_4.12.0-1_amd64.deb wazuh-docker
chris@lst-thread-intel-risk:~$ cd wazuh-docker/
chris@lst-thread-intel-risk:~/wazuh-docker$ cd single-node/
chris@lst-thread-intel-risk:~/wazuh-docker/single-node$ date; docker compose pause
Wed May 26 11:05:24 AM WIB 2026
WARN[0000] /home/chris/wazuh-docker/single-node/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Pausing 2/2
✓ Container single-node-wazuh.manager-1 Paused 0.0s
✓ Container single-node-wazuh.indexer-1 Paused 0.0s
  
```

Lampiran 8 Notifikasi Wazuh Sentinel Muncul 25 Detik Setelah Paused (11:05:24 ~ 11:05:49 WIB) (NF.1.3.)

SOAR Alerting
CRITICAL HEALTH ALERT

Component Offline:
Wazuh Indexer - Wazuh Server C

Failure Details:
Connection timed out. The server is unreachable or slow.

Detected At: 2026-05-20 11:05:49 WIB

```

RX errors 0 dropped 0 overruns 0 frame 0
TX packets 14113617 bytes 4749317261 (4.7 GB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

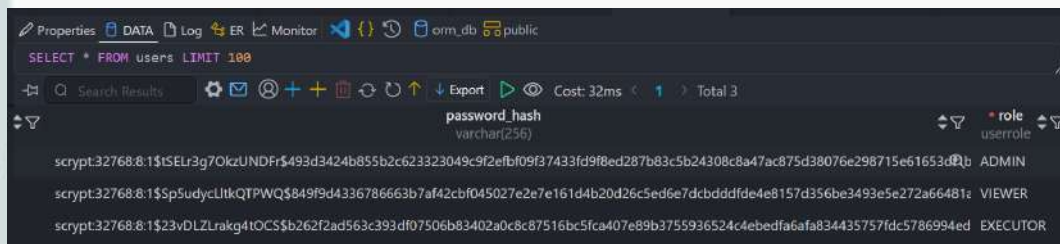
chris@lst-thread-intel-risk:~$ ls
lamis-docker ldap-auto opencti wazuh-agent_4.12.0-1_amd64.deb wazuh-docker
chris@lst-thread-intel-risk:~$ cd wazuh-docker/
chris@lst-thread-intel-risk:~/wazuh-docker$ cd single-node/
chris@lst-thread-intel-risk:~/wazuh-docker/single-node$ date; docker compose pause
Wed May 26 11:05:24 AM WIB 2026
WARN[0000] /home/chris/wazuh-docker/single-node/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Pausing 2/2
✓ Container single-node-wazuh.manager-1 Paused
✓ Container single-node-wazuh.indexer-1 Paused
chris@lst-thread-intel-risk:~/wazuh-docker/single-node$
  
```

Lampiran 9 Field Password Terenkripsi AES dan Berbentuk Ciphertext (NF.2.1)

id	name	url	username	encrypted_password	is_active
2	Ubuntu 2	https://192.168.111.7:9200	admin	gAAAAABo9ddT1dAX10ZbtayicgDTTea-FNTa0SmA3	true
5	Tama Wazuh	https://192.168.111.5:9200	admin	gAAAAABpIWY_7z5gcZNI6UvABow4yXPDVVOjQi	true
6	Chris Indexer	https://192.168.111.6:9200	admin	gAAAAABpoQ0fk-tu4RK0cNaCl69zk1UK5Ce0cnvK	true
1	Honeypot	https://192.168.111.2:9200	admin	gAAAAABo9bRpaBRASKL40cPZBhcW79zeReavenf	true

Hak Cipta :

- Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
 - Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
- Dilarang mengemukakan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Lampiran 10 *Password Field* untuk User Telah Dihash (NF.2.2)


Properties DATA Log ER Monitor orm_db public

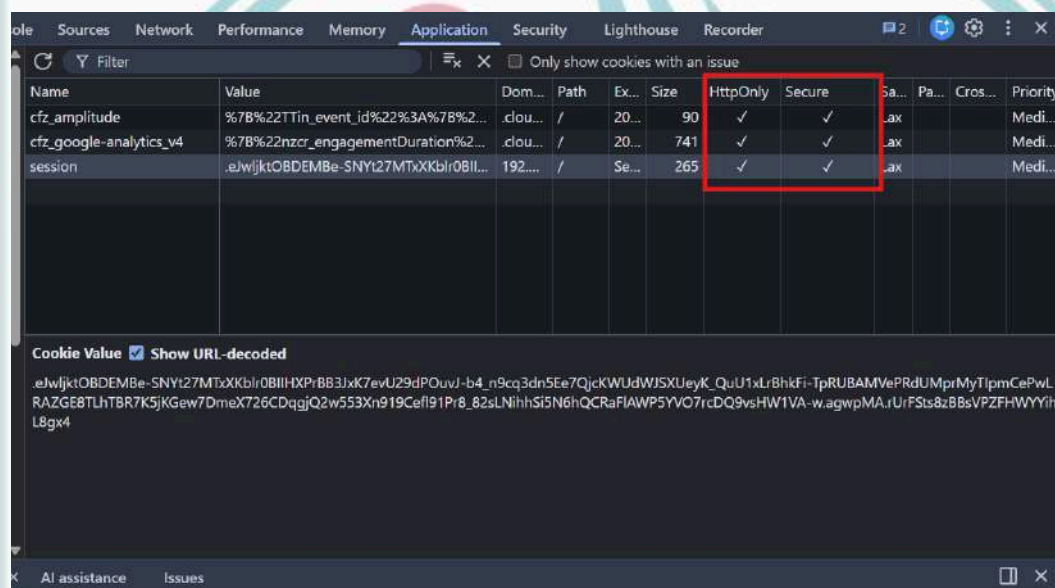
SELECT * FROM users LIMIT 100

Search Results

password_hash varchar(256)

Cost: 32ms 1 Total 3

password_hash	role	userrole
scrypt:32768:8:1\$tSElr3g7OkzUNDFr\$493d3424b855b2c623323049c9f2efbf09f37433fd9f8ed287b83c5b24308c8a47ac875d38076e298715e61653d0b	ADMIN	ADMIN
scrypt:32768:8:1\$SpSudyclUlkQTPWQ\$849f9d4336786663b7af42cbf045027e2e7e161d4b20d26c5ed6e7dcbdddfde4e8157d356be3493e5e272a66481a	VIEWER	VIEWER
scrypt:32768:8:1\$23vDLZLrakg4tOCS\$b262f2ad563c393df07506b83402a0c8c87516bc5fca407e89b3755936524c4ebdfaf6af834435757fde5786994ed	EXECUTOR	EXECUTOR

Lampiran 11 Bukti Pengujian Cookies HTTP Sudah *HttpOnly* dan *Secure* (NF.2.3)


ole Sources Network Performance Memory Application Security Lighthouse Recorder

Filter Only show cookies with an issue

Name	Value	Dom...	Path	Ex...	Size	HttpOnly	Secure	sa...	Pa...	Cros...	Priority
cfz_amplitude	%7B%22Ttin_event_id%22%3A%7B%2...	.clou...	/	20...	90	✓	✓	.ax			Medi...
cfz_google-analytics_v4	%7B%22nzc_engagementDuration%2...	.clou...	/	20...	741	✓	✓	.ax			Medi...
session	.ejwljktOBDEMBE-SNYt27MTxXKblr0BIL...	192...	/	Se...	265	✓	✓	.ax			Medi...

Cookie Value Show URL-decoded

.ejwljktOBDEMBE-SNYt27MTxXKblr0BILHXPrBB3jxK7evU29dPOuwJ-b4_n9cq3dn5Ee7QjcKWUdWJSXUeyK_QuU1xLrBhkFi-TpRUBAMVePRdUMprMyTpmCePwLRAZGE8TLhTBR7K5jKGew7DmeX726CDqgjQ2w553Xn919Cefl91Pr8_82sLNihhS15N6hQCRaFlAWP5YVO71rcDQ9vsHW1VA-w.agwpMA.iUrFSst8zBBsVPZFHWYYihL8gx4

AI assistance Issues



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

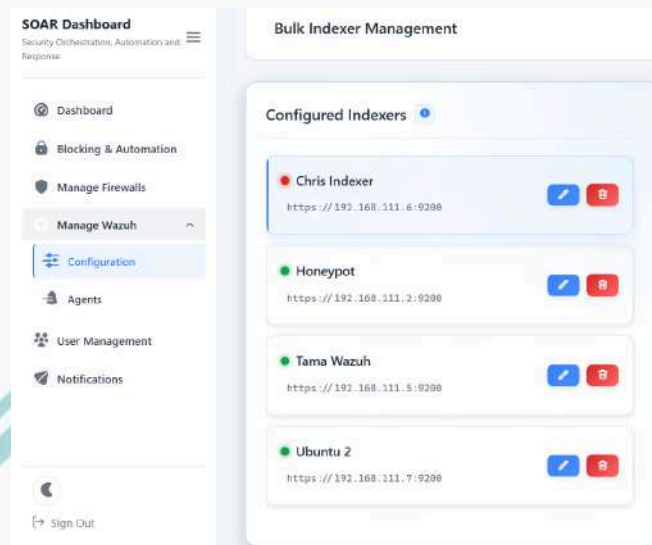
NEGERI
JAKARTA



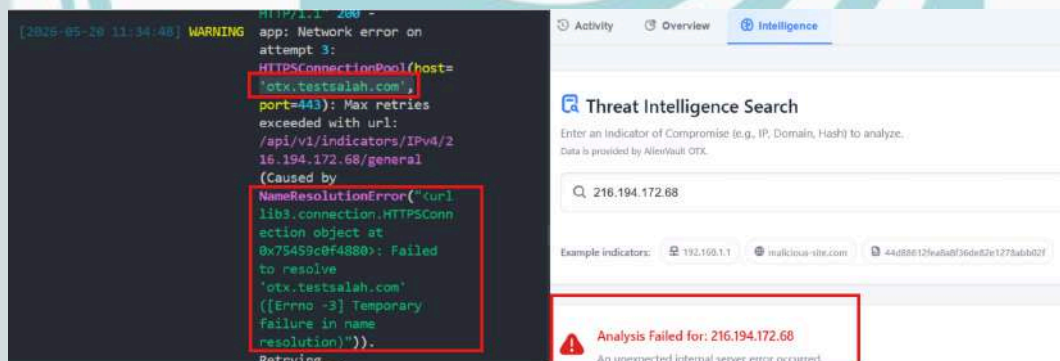
Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengemukakan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Lampiran 12 Kegagalan Salah Satu *Node* Wazuh Membuat Sistem Masih Berjalan (NF.3.1)



Lampiran 13 Isolasi *Error* API *Alienvault* (NF.3.2)





Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Lampiran 14 Tampilan UI es Resiliensi *Scheduler* dengan Menyalahkan *Token* Bot (NF.3.3)

Lampiran 15 Tampilan Log Tes Resiliensi *Scheduler* dengan Menyalahkan *Token* Bot (NF.3.3)

```

2026-05-20 11:54:06] ERROR HTTP/1.1" 200 -
app: HTTP Error sending to
-4933498946, 404 -
{"ok":false,"error_code":404,"description":"Not Found"}
INFO werkzeug: 127.0.0.1 - - [20/May/2026 11:54:06]
"[35m[1mPOST /notifications/telegram/test HTTP/1.1[2m" 500 -
2026-05-20 11:54:07] ERROR app: Network error during
INFO app: Background task finished.
INFO app: Successfully cached summary stats for 58985 alerts.
INFO app: Successfully recalculated and cached overview statistics.
INFO app: Autoblocking is globally disabled. Skipping processing.

```

Lampiran 16 Data Raw *Timestamp* Pengukuran Waktu Penyelesaian (*Time on Task*)

ID Aksi	Metode Eksisting (Manual)			Metode Sistem Usulan (Hub Terpusat)		
	Waktu Mulai (<i>Start</i>)	Waktu Selesai (<i>End</i>)	Total Waktu	Waktu Mulai (<i>Start</i>)	Waktu Selesai (<i>End</i>)	Total Waktu
1	12:06:44	12:07:47	63 detik	11:57:47	11:58:24	37 detik
2	10:53:19	10:54:33	74 detik	10:58:43	10:59:08	25 detik
3	11:11:48	11:12:46	58 detik	11:15:10	11:15:30	20 detik



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

4	11:24:54	11:25:36	42 detik	11:29:44	11:29:56	12 detik
5	11:40:32	11:40:57	25 detik	11:41:09	11:41:26	17 detik
6	12:01:08	12:01:46	38 detik	12:03:40	12:04:00	20 detik
7	15:02:01	15:02:40	39 detik	15:08:22	15:08:36	14 detik
8	15:35:41	15:36:47	66 detik	11:38:33	11:38:52	19 detik
9	15:56:58	15:57:42	44 detik	16:01:30	16:01:55	25 detik
10	16:17:51	16:18:58	67 detik	16:26:16	16:26:42	26 detik

Lampiran 17 Data Video Recording yang Tersedia untuk Verifikasi Uji Efisiensi Operasional

Name	Date	Type	Size	Length
AKSI 1 - MANUAL.mkv	01/06/2026 12:07	MKV File	3.030 KB	00:01:06
AKSI 1 - USULAN.mkv	01/06/2026 11:58	MKV File	2.752 KB	00:00:44
AKSI 2 - MANUAL.mkv	02/06/2026 10:54	MKV File	2.520 KB	00:01:18
AKSI 2 - USULAN.mkv	02/06/2026 10:59	MKV File	1.601 KB	00:00:29
AKSI 3 - MANUAL.mkv	02/06/2026 11:12	MKV File	1.921 KB	00:00:43
AKSI 3 - USULAN.mkv	02/06/2026 11:15	MKV File	1.650 KB	00:00:23
AKSI 4 - MANUAL.mkv	02/06/2026 11:25	MKV File	4.736 KB	00:00:46
AKSI 4 - USULAN.mkv	02/06/2026 11:29	MKV File	1.508 KB	00:00:17
AKSI 5 - MANUAL.mkv	02/06/2026 11:40	MKV File	4.375 KB	00:00:28
AKSI 5 - USULAN.mkv	02/06/2026 11:41	MKV File	3.972 KB	00:00:20
AKSI 6 - MANUAL.mkv	02/06/2026 12:01	MKV File	3.470 KB	00:00:43
AKSI 6 - USULAN.mkv	02/06/2026 12:04	MKV File	3.536 KB	00:00:27
AKSI 7 - MANUAL.mkv	02/06/2026 15:02	MKV File	1.316 KB	00:00:47
AKSI 7 - USULAN.mkv	02/06/2026 15:08	MKV File	1.861 KB	00:00:18
AKSI 8 - MANUAL.mkv	02/06/2026 15:36	MKV File	5.169 KB	00:01:15
AKSI 8 - USULAN.mkv	02/06/2026 15:38	MKV File	2.030 KB	00:00:22
AKSI 9 - MANUAL.mkv	02/06/2026 15:57	MKV File	1.926 KB	00:00:51
AKSI 9 - USULAN.mkv	02/06/2026 16:01	MKV File	3.376 KB	00:00:29
AKSI 10 - MANUAL.mkv	02/06/2026 16:18	MKV File	1.662 KB	00:01:14
AKSI 10 - USULAN.mkv	02/06/2026 16:26	MKV File	4.617 KB	00:00:42