ISSN (Online) 2747-0563
Volume 6 Tahun 2026

Integritas JSON pada Otomatisasi Firewall Filter MikroTik Menggunakan n8n dan LLM Qwen2.5-Coder-7B

Muhammad Rafi M N[1], Achmad Junaidi[2*], Eva Yulia Puspaningrum[3]

[1,2,3] Informatika, Universitas Pembangunan Nasional "Veteran" Jawa Timur
[1]22081010201@student.upnjatim.ac.id
[3]evapuspaningrum.if@upnjatim.ac.id

*Corresponding author email: achmadjunaidi.if@upnjatim.ac.id

***Abstrak*— Otomatisasi manajemen kebijakan *firewall* MikroTik menggunakan *Large Language Model* (LLM) menawarkan solusi adaptif untuk mengatasi proses administrasi manual yang memakan waktu dan rentan terhadap *human error*. Namun, penerapan *base model* kecerdasan buatan sering kali memunculkan celah keamanan baru akibat fenomena "halusinasi" parameter. Hal ini dipicu oleh ketidakmampuan model berbasis semantik dalam mematuhi batasan leksikal yang kaku pada sintaks RouterOS MikroTik. Penelitian ini merancang dan mengevaluasi arsitektur sistem otomatisasi berbasis instruksi bahasa natural (bahasa Indonesia) yang diorkestrasi menggunakan platform n8n, model Qwen2.5-Coder-7B-Instruct melalui vLLM, dan basis data Supabase. Pengujian dilakukan terhadap 30 skenario *test case* yang terdistribusi secara seimbang pada rantai lalu lintas *input*, *forward*, dan *output*, serta tindakan *accept* dan *drop*. Hasil pengujian kuantitatif menunjukkan tingkat keberhasilan sebesar 90% dalam menghasilkan format luaran JSON yang valid secara sintaksis dan akurat secara leksikal. Meskipun demikian, analisis kualitatif terhadap 10% kegagalan mengungkap adanya misinterpretasi model terhadap arah paket lalu lintas dan kegagalan dalam menangkap instruksi negasi. Berdasarkan temuan tersebut, penerapan teknik *Retrieval-Augmented Generation* (RAG) dan optimasi *Prompt Engineering* terbukti krusial untuk menjaga presisi parameter alamat IP. Sebagai langkah mitigasi lanjutan, metode *fine-tuning* direkomendasikan untuk menanamkan pemahaman leksikal bawaan pada model guna meningkatkan keandalan sistem pada lingkungan jaringan aktual.**

Kata Kunci— **Otomatisasi Jaringan, Firewall MikroTik, Large Language Model, n8n, Retrieval-Augmented Generation.**

I. PENDAHULUAN

Keamanan data sebuah organisasi sangat bergantung pada perlindungan yang kuat terhadap aplikasi dan infrastruktur jaringannya [1]. Seiring dengan pesatnya transformasi digital, ancaman siber menjadi kian kompleks sehingga menuntut pertahanan jaringan yang adaptif dan presisi. Meskipun infrastruktur jaringan sering kali dianggap solid dengan perlindungan firewall tingkat dasar yang baik, celah kerentanan tingkat menengah (medium risk) sering muncul di lapisan aplikasi dan manajemen kebijakan [1]. Referensi [1] menegaskan bahwa kerentanan konfigurasi yang tidak optimal pada infrastruktur dapat menjadi pintu masuk bagi eksploitasi, sehingga validasi keamanan infrastruktur secara komprehensif menjadi sangat esensial. Selama ini, tantangan terbesar dalam memitigasi celah tersebut adalah proses administrasi dan manajemen kebijakan firewall yang masih banyak dilakukan secara manual, di mana proses ini memakan waktu dan sangat rentan terhadap human error.

Untuk mengatasi tantangan administrasi manual tersebut, integrasi kecerdasan buatan melalui Large Language Models (LLM) seperti Qwen2.5-Coder-7B-Instruct yang dipadukan dengan platform workflow n8n menawarkan potensi solusi otomatisasi yang menjanjikan. Sistem ini diharapkan mampu menerjemahkan bahasa natural menjadi perintah konfigurasi otomatis pada perangkat firewall seperti MikroTik. Akan tetapi, potensi solusi otomatisasi ini perlu dikritisi karena memiliki risiko kegagalan teknis yang tinggi saat berhadapan dengan frasa teknis jaringan yang sangat spesifik. Hal ini didukung oleh temuan [2] yang menyoroti bahwa model bahasa AI (khususnya dalam kondisi baseline) sering kali memiliki keterbatasan dalam menangani istilah teknis spesifik dan konteks domain tertentu dibandingkan dengan metode konvensional [2].

Dalam operasional firewall MikroTik, aturan kebijakan mengikat pada sintaks yang pasti dan tidak menoleransi ambiguitas. Sebagaimana diargumenkan oleh [2], metode pendekatan leksikal konvensional (seperti aturan baku pada MikroTik yang mengandalkan kecocokan istilah secara eksak) masih jauh lebih unggul dalam menjaga presisi istilah. Sebaliknya, model semantik AI cenderung melakukan generalisasi berlebih terhadap makna informasi, sehingga berisiko melewatkan parameter atau frasa penting yang sangat krusial dalam konteks teknis [2]. Keterbatasan AI dalam memahami aturan leksikal yang kaku ini bermuara pada masalah spesifik berupa "halusinasi" LLM, di mana model sering kali memproduksi format parameter terstruktur (seperti JSON) yang cacat, tidak konsisten, atau mengarang sintaks yang tidak valid saat dieksekusi oleh n8n.

Untuk mengatasi kelemahan mendasar dari *base model* tersebut, diperlukan intervensi rekayasa yang terstruktur melalui kombinasi *Retrieval-Augmented Generation* (RAG) dan *Prompt Engineering* [3]. Integrasi RAG memainkan peran krusial dengan menyuntikkan basis pengetahuan eksternal serta data topologi jaringan secara riil (seperti parameter operasional terkini) ke dalam konteks model, yang secara efektif membumikan penalaran AI pada kondisi faktual dan mereduksi risiko halusinasi secara signifikan [3], [4]. Secara bersamaan, penerapan *Prompt Engineering* bertugas memaksakan kepatuhan model terhadap instruksi format secara deterministik, guna memastikan bahwa luaran yang dihasilkan

ISSN (Online) 2747-0563
Volume 6 Tahun 2026

selaras dengan sintaks JSON yang valid serta mematuhi aturan leksikal perangkat jaringan yang kaku [3], [5]. Perpaduan kedua metode ini secara strategis dipilih karena menawarkan pendekatan yang jauh lebih dinamis, terukur, dan efisien dibandingkan keharusan melakukan *fine-tuning* model secara menyeluruh; di mana *fine-tuning* tidak hanya menuntut ketersediaan data latih dan sumber daya komputasi yang masif, tetapi juga lebih lambat dalam beradaptasi, sedangkan pendekatan RAG dan *prompting* memungkinkan administrator untuk menyesuaikan perubahan jaringan seketika hanya dengan memperbarui basis pengetahuan eksternalnya tanpa perlu melatih ulang parameter model [3], [4].

Dalam era transformasi digital dan peningkatan kompleksitas infrastruktur jaringan, orkestrasi *workflow* menggunakan platform *low-code* seperti n8n telah menjadi pendekatan esensial untuk mengeksekusi *pipeline* otomatisasi secara efisien [6]. Implementasi n8n memungkinkan integrasi tanpa batas antara pemantauan log keamanan (seperti sistem syslog pada MikroTik RouterOS) dengan aksi mitigasi otonom melalui antarmuka pemrograman aplikasi (API), yang secara signifikan mereduksi latensi respons dibandingkan intervensi administratif manual [7]. Skema orkestrasi *self-hosted* ini tidak hanya menyederhanakan konfigurasi yang rumit, tetapi juga memberikan performa yang deterministik serta *overhead* sistem yang sangat rendah, menjadikannya fondasi yang kokoh untuk lingkungan operasional yang mengedepankan kontrol privasi dan skalabilitas penuh [6].

Namun, untuk menggeser paradigma dari sekadar skrip berbasis aturan (rule-based) menuju otomatisasi cerdas berkonsep *Intent-Based Networking* (IBN), platform orkestrasi harus diintegrasikan dengan *Large Language Models* (LLM) untuk menginterpretasikan instruksi pengguna [3]. Dalam alur kerja ini, menjaga robasitas keluaran skema JSON dari LLM menjadi sangat krusial; cacat sintaksis atau halusinasi format sekecil apa pun dapat berakibat fatal pada terjadinya *misconfiguration* perangkat [3]. Tantangan terbesar bagi integrasi AI dalam jaringan adalah kemampuannya menangani ambiguitas bahasa natural—seperti frasa kolokial, instruksi tidak baku, atau instruksi multi-domain—yang sering kali menyebabkan model gagal memetakan niat (*intent*) ke dalam kebijakan jaringan (policy) secara akurat [3].

Oleh karena itu, penelitian ini sangat mendesak untuk dilakukan guna mengukur error rate dari integritas data JSON yang dihasilkan pada integrasi n8n dan LLM Qwen. Evaluasi ini bertujuan untuk memastikan kelayakan operasional dari pemanfaatan AI dalam menerjemahkan kebijakan jaringan, sehingga proses otomatisasi kebijakan firewall MikroTik tersebut tidak justru menciptakan celah keamanan baru akibat instruksi sistem yang berhalusinasi atau terdistorsi.

II. METODOLOGI PENELITIAN

Bab ini menguraikan tahapan dan pendekatan metodologis yang digunakan dalam merancang serta mengevaluasi sistem otomatisasi kebijakan *firewall* MikroTik. Pembahasan di dalam bab ini akan difokuskan pada perencanaan ruang lingkup pengujian model Qwen2.5-Coder-7B-Instruct, penyusunan skenario *test case* berbasis bahasa natural, hingga penentuan metrik analisis yang diterapkan untuk mengukur tingkat keberhasilan integrasi sistem pada platform n8n.

A. Perencanaan dan Penentuan Ruang Lingkup

Penelitian ini difokuskan pada perancangan dan evaluasi sistem otomatisasi manajemen kebijakan firewall MikroTik yang diorkestrasi menggunakan platform workflow n8n. Model kecerdasan buatan utama yang digunakan dalam penelitian ini adalah Qwen2.5-Coder-7B-Instruct.

Ruang lingkup penelitian ini secara spesifik dibatasi pada penerjemahan instruksi bahasa natural menjadi parameter kebijakan firewall filter MikroTik. Parameter krusial yang menjadi batasan pengujian difokuskan pada penentuan rantai lalu lintas (chain) dan tindakan (action) dengan definisi operasional. Berikut untuk rantai lalu lintas (*Chain*):

1) *Input*: Digunakan untuk memproses paket yang memasuki router (entering the router) melalui salah satu antarmuka di mana alamat IP tujuannya adalah salah satu alamat IP milik router itu sendiri. Paket yang hanya menumpang lewat (passing through) router tidak akan diproses oleh aturan-aturan yang ada di dalam chain input ini.

2) *Forward*: Digunakan untuk memproses paket yang melewati router (passing through the router) dari satu antarmuka jaringan ke antarmuka jaringan yang lain.

3) *Output*: Digunakan untuk memproses paket yang asalnya dari router itu sendiri (originating from the router) dan keluar meninggalkannya melalui salah satu antarmuka. Sebagaimana chain input, paket yang hanya menumpang lewat router tidak akan diproses oleh aturan-aturan pada chain output

Selanjutnya untuk definisi tindakan (*Action*):

1) *Accept*: Menerima paket (*accept the packet*). Jika sebuah paket cocok dengan aturan ini, maka paket tersebut akan langsung diterima dan tidak akan diteruskan lagi untuk diperiksa pada aturan firewall berikutnya.

2) *Drop*: Menolak atau membuang paket secara diam-diam (*silently drop the packet*).

Selain batasan leksikal di atas, hal utama yang diuji adalah kemampuan LLM dalam memproduksi output JSON yang harus memiliki field-field spesifik secara terstruktur dan valid. Model dituntut untuk bisa memetakan instruksi ke dalam field *summary, analysis, risk, intent, selected_tool, network_device*, beserta susunan konfigurasi pada array *commands*.

Sebagai contoh implementasi, apabila model diberikan instruksi untuk memutuskan koneksi dari IP tertentu menuju IP *router* Cilacap, model harus secara presisi mengeluarkan *output* JSON seperti pada Gbr. 1.

ISSN (Online) 2747-0563
Volume 6 Tahun 2026


```
{
  "output": {
    "summary": "Drop connection from IP
10.20.30.50 to IP router Cilacap 10.20.30.1",
    "analysis": "The instruction requires
blocking traffic from a specific source IP
address (10.20.30.50) to a destination IP
address (10.20.30.1). This can be achieved by
adding a firewall rule on the router with the
IP address 10.20.30.1.",
    "risk": "High",
    "intent": "Block incoming traffic from
10.20.30.50 to 10.20.30.1",
    "selected_tool": "firewall_filter",
    "network_device": "10.20.30.1",
    "commands": [
      {
        "type": "add",
        "path": "/ip/firewall/filter",
        "payload": {
          "chain": "input",
          "src-address": "10.20.30.50",
          "dst-address": "10.20.30.1",
          "action": "drop"
        }
      }
    ]
  }
}
```


Gbr. 1 Contoh *output* yang harus dikeluarkan oleh model Qwen2.5-Coder-7B-Instruct.

B. Alur dan Infrastruktur Pengujian

Pengujian dalam penelitian ini dirancang menggunakan alur sistem terintegrasi yang melibatkan beberapa platform. Proses pengujian diawali dari antarmuka pengguna berbasis situs web (website) yang berfungsi sebagai wadah untuk menerima masukan (input) berupa instruksi jaringan (intent) dari pengguna. Informasi instruksi dari situs web tersebut kemudian ditransmisikan menuju platform orkestrasi workflow n8n. Di dalam n8n, seluruh informasi masukan beserta konteks tambahan dikelola dan dikompilasi menjadi sebuah prompt terstruktur.

Setelah prompt terbentuk, n8n akan meneruskannya ke mesin inferensi (inference engine). Mesin inferensi yang digunakan dalam penelitian ini beroperasi di atas infrastruktur komputasi awan RunPod, yang menjalankan framework vLLM sebagai peladen (server) untuk mengeksekusi model Large Language Model (LLM) Qwen2.5-Coder-7B-Instruct. n8n melakukan pemanggilan secara terprogram ke endpoint RunPod tersebut, dan setelah LLM selesai melakukan inferensi, hasil luaran (output) dikembalikan lagi ke platform n8n. Pada tahap ini, n8n melakukan pengolahan lanjutan (post-processing) untuk mengekstraksi dan memvalidasi luaran tersebut agar benar-benar sesuai dengan spesifikasi dan batasan format JSON yang telah ditetapkan pada skema desain luaran.

Seluruh data transaksi pengujian, yang meliputi instruksi awal (original prompt), hasil keluaran mentah dari LLM (raw output), dan parameter terkait lainnya, direkam secara otomatis dan disimpan ke dalam pangkalan data Supabase. Untuk tahap penilaian validitas output, pengujian tidak menggunakan mekanisme grader otomatis. Penentuan apakah luaran JSON tersebut telah sesuai dengan format ketentuan leksikal MikroTik dan memenuhi harapan teknis dari intent pengguna pada setiap test case dilakukan secara manual oleh peneliti. Evaluasi manual ini dilakukan secara ketat dengan berlandaskan pada komparasi luaran terhadap pedoman dokumentasi resmi MikroTik RouterOS, sehingga anomali logika atau "halusinasi" sekecil apa pun dari model dapat diidentifikasi secara akurat.

C. Test Case Pengujian

Pengujian kapabilitas model Large Language Model (LLM) dilakukan menggunakan serangkaian skenario test case berupa perintah berbasis bahasa natural (bahasa Indonesia). Untuk memastikan pengujian mencakup seluruh batasan parameter leksikal yang telah ditentukan, dirancang total 30 perintah (instruksi jaringan) yang didistribusikan secara proporsional.

Distribusi pengujian dibagi secara merata ke dalam tiga jenis rantai lalu lintas (*chain*) utama pada *firewall filter* MikroTik. Setiap *chain*, yaitu *input*, *forward*, dan *output*, masing-masing akan diuji menggunakan 10 perintah spesifik. Lebih lanjut, untuk mengukur tingkat keakuratan model dalam mengidentifikasi tindakan yang tepat, 10 perintah pada masing-masing *chain* tersebut diklasifikasikan kembali menjadi dua kelompok tindakan (*action*) yang berimbang. Sejumlah 5 perintah dirancang untuk mensimulasikan instruksi pemblokiran paket jaringan (*action drop*), dan 5 perintah lainnya dirancang untuk mensimulasikan instruksi pemberian izin akses lalu lintas jaringan (*action accept*). Rincian distribusi *test case* pengujian tersebut dapat dilihat pada Tabel I di bawah.

TABEL I
RINCIAN DISTRIBUSI TEST CASE

Chain	*Action*	Jumlah Perintah	Total Perintah
Input	*Drop*	5	10
	Accept	5	
Forward	*Drop*	5	10
	Accept	5	
Output	*Drop*	5	10
	Accept	5	
Total Keseluruhan *Test Case*			30

Berdasarkan Tabel I, rancangan pengujian mengadopsi pendekatan distribusi seimbang (*balanced distribution*) dengan total 30 perintah. Keseimbangan ini sengaja diterapkan untuk mengeliminasi bias evaluasi model, sehingga performa model Qwen2.5-Coder-7B-Instruct dapat diukur secara adil pada

ISSN (Online) 2747-0563
Volume 6 Tahun 2026

setiap kondisi rantai lalu lintas (*chain*). Penetapan 10 perintah pada masing-masing *chain* (*input, forward, output*) bertujuan untuk menguji sensitivitas model terhadap arah paket (*packet directionality*) yang berbeda secara semantik di MikroTik RouterOS. Sementara itu, pembagian sub-aksi menjadi masing-masing 5 perintah untuk *action drop* dan *accept* ditujukan untuk memvalidasi kemampuan penalaran logis model dalam membedakan instruksi bernada izin (*permissive*) dan instruksi bernada pemblokiran atau negasi (*restrictive*).

Kemudian untuk kode dan pertanyaan lengkap setiap *test case* dijelaskan pada Tabel II.

TABEL II
RINCIAN TEST CASE

Chain	Action	No. TC	Perintah
INPUT	DROP	IN-DR-01	Tolong blokir akses ping dari IP 10.10.20.15 menuju ke router Sidoarjo di 10.10.10.1.
		IN-DR-02	Blokir user dengan IP 10.20.20.100 yang nyoba-nyoba login ke router hAP Cilacap di 10.20.30.1.
		IN-DR-03	Tolong drop koneksi dari IP 10.20.30.50 yang mau masuk ke IP router Cilacap 10.20.30.1.
		IN-DR-04	Tolong putusin akses dari IP tamu 10.30.20.200 yang mau ngakses IP router 10.30.20.1.
		IN-DR-05	Larang IP 10.10.20.99 buat nyentuh atau ngakses router 10.10.10.1 sama sekali.
	ACCEPT	IN-AC-01	Buka dong jalur buat IP 10.30.20.55 supaya bisa ngakses masuk ke router Surabaya (10.30.20.1).
		IN-AC-02	Izinin subnet lokal 10.10.20.0/24 buat masuk ke router pusat Sidoarjo 10.10.10.1.
		IN-AC-03	Accept akses masuk dari jaringan 10.20.10.0/24 menuju ke router 10.20.30.1.
		IN-AC-04	Bolehin admin dari IP 10.10.10.5 buat remote langsung ke router 10.10.10.1.
		IN-AC-05	Buka pintu firewall buat subnet 10.20.30.0/24 biar bisa masuk ke IP router 10.20.30.1.
FORWARD	DROP	FO-DR-01	Matiin traffic dari network 10.20.10.0/24 yang mau nyebrang lewat router Cilacap (10.20.30.1) ke arah network 10.20.20.0/24.
		FO-DR-02	Jangan kasih lewat traffic dari 10.30.10.0/24 yang mau menuju 10.30.20.0/24 di router 10.30.20.1.
		FO-DR-03	Blok koneksi antar cabang nih,

ISSN (Online) 2747-0563
Volume 6 Tahun 2026

Chain	Action	No. TC	Perintah
			dari 10.20.20.0/24 gak boleh lewat router 10.10.10.1 buat menuju 10.30.20.0/24.
		FO-DR-04	Tolong tutup jalur transit di router 10.30.20.1 dari jaringan 10.30.20.0/24 yang mau ke 10.10.20.0/24.
		FO-DR-05	Cegat aja data dari IP 10.20.20.20 yang mau numpang lewat router 10.20.30.1 menuju ke IP 10.10.10.10.
	ACCEPT	FO-AC-01	Bolehin akses dari IP 10.10.10.10 buat menuju ke subnet 10.10.20.0/24 melewati router Main Sidoarjo (10.10.10.1).
		FO-AC-02	Buka akses lewatin router 10.10.10.1 dari IP 10.10.20.5 menuju ke server 10.10.10.10.
		FO-AC-03	Kasih jalur buat IP 10.20.30.2 supaya bisa ngelewatin router 10.20.30.1 menuju ke subnet 10.20.20.0/24.
		FO-AC-04	Buka firewall forward di router 10.10.10.1 biar IP 10.10.20.10 bisa komunikasi

Chain	Action	No. TC	Perintah
			leluasa ke 10.30.20.0/24.
		FO-AC-05	Izinkan traffic forward dari network 10.30.10.0/24 menuju ke 10.30.20.0/24 lewat router 10.30.20.1.
OUTPUT	DROP	OU-DR-01	Seting router 10.10.10.1 biar gak bisa ngirim paket apa pun keluar ke jaringan 10.20.30.0/24.
		OU-DR-02	Tahan paket dari router Sidoarjo 10.10.10.1 yang nyoba mau keluar ke IP 10.30.20.2.
		OU-DR-03	Blokir output dari router 10.20.30.1 yang asalnya dari router itu sendiri buat menuju ke IP 10.30.20.5.
		OU-DR-04	Set biar router hAP Lite Surabaya (10.30.20.1) gak bisa nge-ping atau ngirim data keluar ke 10.20.30.0/24.
		OU-DR-05	Blok semua paket yang keluar dari router 10.10.10.1 menuju ke target IP spesifik 10.20.20.50.
	ACCEPT	OU-AC-01	Tolong kasih izin router Cilacap (10.20.30.1) buat konek keluar menuju ke IP 10.10.10.10.

ISSN (Online) 2747-0563
Volume 6 Tahun 2026

Chain	Action	No. TC	Perintah
		OU-AC-02	Biarkan router 10.30.20.1 ngirim data dari sistemnya sendiri ke subnet 10.10.10.0/24.
		OU-AC-03	Izinkan trafik keluar dari router Sidoarjo 10.10.10.1 buat menuju ke network 10.20.20.0/24.
		OU-AC-04	Tolong allow koneksi keluar dari router 10.20.30.1 ke arah IP server 10.10.10.10.
		OU-AC-05	Izinkan router 10.30.20.1 buat ngirim trafik langsung ke IP 10.10.10.1 dari routernya.

Tabel II menunjukkan variasi gaya bahasa yang digunakan dalam merumuskan *intent* jaringan, mulai dari bahasa formal hingga bahasa sehari-hari yang bersifat kolokial (seperti penggunaan kata *"nyoba-nyoba"*, *"putusin"*, *"biar bisa komunikasi leluasa"*). Penggunaan variasi leksikal bahasa Indonesia non-formal ini merupakan representasi dari kondisi riil di lapangan, di mana administrator jaringan atau pengguna awam sering kali memasukkan instruksi yang tidak baku. Pengujian dengan variasi ini sangat krusial untuk mengukur sejauh mana model *open-source* berukuran 7B mampu mempertahankan akurasi leksikal parameter rigid MikroTik (seperti *src-address* dan *dst-address*) ketika dihadapkan pada fleksibilitas semantik bahasa natural manusia.

D. Analisis dan Rekomendasi

Tahapan analisis hasil dan perumusan rekomendasi merupakan fase akhir dalam metodologi penelitian ini. Evaluasi terhadap luaran (*output*) model Qwen2.5-Coder-7B-Instruct dilakukan secara komprehensif dengan menggabungkan dua pendekatan utama, yaitu analisis kuantitatif dan analisis kualitatif.

1) *Analisis Kuantitatif*: Analisis kuantitatif difokuskan pada pengukuran performa keandalan sistem melalui metrik Tingkat Keberhasilan (Success Rate) dan Tingkat Kesalahan (*Error Rate*). Pengukuran ini dilakukan terhadap 30 skenario *test case* yang telah dieksekusi. Parameter keberhasilan didasarkan pada nilai boolean is_success yang direkam dalam pangkalan data Supabase. Suatu eksekusi dinyatakan berhasil (true) apabila model mampu memproduksi struktur JSON yang utuh dan akurat secara leksikal, di mana seluruh *field* krusial (khususnya penentuan nilai *chain*, *action*, dan alamat IP pada elemen *commands*) sesuai dengan pedoman standar MikroTik dan selaras dengan maksud instruksi pengguna.

2) Analisis Kualitatif Analisis kualitatif dilakukan dengan membedah secara mendalam data log yang diklasifikasikan gagal (is_success: false). Peneliti akan melakukan inspeksi manual secara detail dengan menyandingkan parameter *original_prompt* (instruksi awal dari pengguna) dengan *raw_llm_output* (hasil generasi JSON dari AI). Tinjauan kualitatif ini bertujuan untuk mengidentifikasi pola kesalahan atau anomali spesifik, di antaranya:

- **Kesalahan Logika *Chain***: Mengevaluasi apakah model gagal membedakan konteks arah lalu lintas, seperti tertukarnya penggunaan chain input (paket menuju *router*) dengan *chain forward* (paket melintasi router).
- **Kesalahan Penerjemahan *Action***: Menelisik apakah model gagal menangkap makna negasi dalam instruksi, sehingga keliru memberikan *action accept* pada instruksi yang seharusnya memblokir (*drop*).
- **Fenomena Halusinasi**: Mengidentifikasi kemungkinan di mana model mengarang parameter baru atau merusak format skema JSON yang telah ditetapkan.

3) *Perumusan Rekomendasi* Berdasarkan temuan yang diperoleh dari hasil analisis kuantitatif dan kualitatif, tahapan selanjutnya adalah merumuskan rekomendasi perbaikan arsitektural. Rekomendasi ini ditujukan untuk memitigasi tingkat kegagalan (*error rate*) pada iterasi pengembangan sistem selanjutnya. Fokus utama dari rekomendasi mencakup:

- **Eksplorasi Metode *Fine-Tuning***: Sebagai solusi tingkat lanjut untuk mengurangi ketergantungan pada *System Prompt* yang panjang, direkomendasikan untuk melakukan *fine-tuning* pada model dasar (Qwen2.5-Coder-7B-Instruct). Dengan melatih ulang model menggunakan dataset spesifik berisi pasangan intent jaringan (berbahasa Indonesia) dan format JSON MikroTik yang valid, model akan memiliki insting bawaan untuk mengenali instruksi pengguna dan memetakan batasan leksikal secara mandiri tanpa memerlukan optimasi prompt yang berlebihan.

ISSN (Online) 2747-0563
Volume 6 Tahun 2026

- **Optimalisasi *Prompt Engineering***: Selama *fine-tuning* belum diimplementasikan, penyesuaian instruksi dasar pada n8n tetap perlu diperketat agar instruksi mengenai batasan leksikal MikroTik dapat dipahami oleh LLM secara lebih deterministik dan kaku.
- **Penyempurnaan Konteks RAG (*Retrieval-Augmented Generation*)**: Peningkatan logika pemrosesan JavaScript pada n8n untuk memastikan bahwa daftar perangkat dan alamat IP yang disuapkan ke dalam *prompt* LLM sudah terdeduplikasi dan sepenuhnya relevan dengan topologi aktual.
- **Pengetatan Parameter JSON**: Evaluasi terhadap parameter pengunci response_format pada vLLM untuk memastikan model tidak dapat keluar dari skema struktur objek yang diizinkan.

III. *HASIL DAN PEMBAHASAN*

Bab ini memaparkan secara komprehensif hasil pengujian dan pembahasan terkait implementasi sistem otomatisasi manajemen kebijakan *firewall* MikroTik yang digerakkan oleh kecerdasan buatan. Selaras dengan tahapan metodologi yang telah dirancang pada Bab II, evaluasi dilakukan terhadap 30 skenario *test case* instruksi bahasa natural yang dieksekusi melalui *website* kemudian diteruskan kedalam platform orkestrasi n8n dan diproses oleh mesin inferensi Qwen2.5-Coder-7B-Instruct. Sistematika pembahasan pada bab ini dibagi menjadi beberapa bagian utama, diawali dengan penyajian data empiris secara kuantitatif untuk mengukur tingkat keberhasilan (*success rate*) dari integritas format JSON yang dihasilkan oleh model. Selanjutnya, dilakukan analisis kualitatif mendalam untuk membedah anomali, mis-interpretasi leksikal, dan fenomena halusinasi pada data yang gagal. Bab ini diakhiri dengan evaluasi perbandingan metode serta perumusan rekomendasi arsitektural meliputi optimalisasi *Prompt Engineering*, penyempurnaan teknik *Retrieval Augmented Generation (RAG)*, hingga penjajakan opsi *fine-tuning* guna meningkatkan keandalan sistem pada lingkungan jaringan aktual.

A. *Hasil Implementasi Sistem*

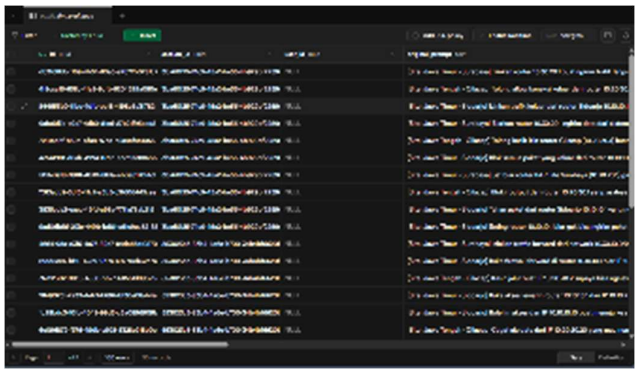
Tahap implementasi merupakan perwujudan dari perancangan arsitektur yang telah disusun pada bab sebelumnya. Sistem otomatisasi ini berhasil diintegrasikan dengan menghubungkan antarmuka *website*, platform orkestrasi n8n, *engine* vLLM pada infrastruktur RunPod, dan basis data Supabase sebagai penyimpan data relasional serta log aktivitas.

Seluruh logika bisnis dan aliran data diatur melalui *workflow* pada platform n8n. Implementasi ini memungkinkan sistem untuk menerima *input* dari pengguna, melakukan pengayaan data (*data enrichment*) melalui teknik RAG, hingga mengirimkan instruksi ke model bahasa. Alur kerja lengkap dari sistem ini dapat dilihat pada Gbr. 2.

Dalam pengembangan *workflow* ini, ditemukan bahwa penggunaan node HTTP Request merupakan metode yang paling optimal untuk melakukan pemanggilan (*inference*) ke model Qwen2.5-Coder-7B-Instruct yang berjalan di atas vLLM. Optimasi ini didasarkan pada beberapa alasan teknis yaitu kendali penuh terhadap *payload*, manajemen *autentikasi*, efisiensi *latensi*.

Gbr. 2 Tampilan *platform* n8n.

Selanjutnya, supabase digunakan sebagai pilar utama dalam penyimpanan data topologi jaringan dan pencatatan setiap interaksi LLM. Struktur tabel yang diimplementasikan mencakup tabel topology_nodes untuk menyimpan identitas perangkat, serta tabel llm_eval_logs untuk merekam data evaluasi. Tampilan antarmuka basis data yang telah terpopulasi dengan data pengujian ditunjukkan pada Gbr. 3



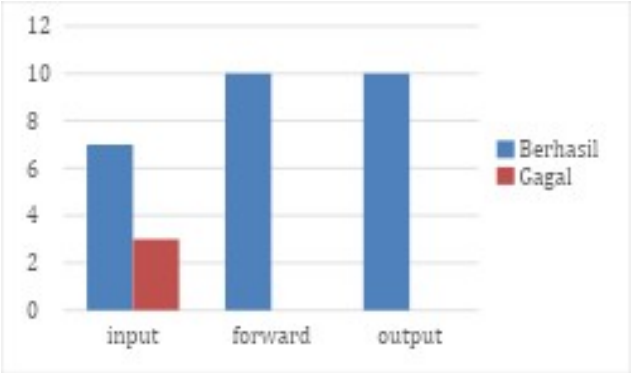
Gbr. 3 Tampilan *database* supabase untuk penyimpanan pengujian.

Setiap baris data dalam Supabase menyimpan rekaman krusial seperti *session ID*, instruksi asli dari pengguna (*original prompt*), dan hasil luaran dari AI (*raw LLM output*). Hal ini memudahkan peneliti dalam melakukan audit teknis dan penilaian manual terhadap integritas data JSON yang dihasilkan, guna memastikan bahwa setiap konfigurasi yang dirumuskan telah mematuhi batasan leksikal MikroTik.

B. *Hasil Pengujian Kuantitatif*

Pengujian telah dilakukan terhadap 30 skenario instruksi bahasa natural yang dikirimkan melalui situs web menuju *workflow* n8n dan diproses oleh model Qwen2.5-Coder-7B-Instruct. Tingkat keberhasilan (*Success Rate*) diukur berdasarkan kemampuan model menghasilkan JSON yang valid secara sintaksis dan akurat secara leksikal MikroTik (terekam sebagai is_success: true pada basis data) hasil dari pengujian dapat dilihat di Gbr. 4.

ISSN (Online) 2747-0563
Volume 6 Tahun 2026



Gbr. 4 Grafik hasil pengujian menggunakan *test case*. Menunjukkan bahwa *chain* input perlu dilakukan optimalisasi prompt dibanding *chain* lainnya.

C. *Analisis Kualitatif Kegagalan*

Meskipun model menunjukkan tingkat keberhasilan yang tinggi, terdapat tiga *test case* yang diklasifikasikan gagal (is_success: false). Berdasarkan inspeksi manual antara *original_prompt* dan *raw_llm_output*, kegagalan tersebut tidak disebabkan oleh rusaknya struktur JSON, melainkan oleh kelemahan semantik AI dalam memahami batasan leksikal MikroTik maupun kurngnya pemahaman terhadap *intent* pengguna. Berikut adalah pembedahan anomali yang ditemukan:

1) *Kesalahan Identifikasi Intent Pengguna:* Pada pengujian nomor IN-DR-05 pengguna sengaja membuat kalimat yang multi-tafsir namun jika dipahami lebih dalam perintah tersebut memerintahkan untuk memblokir suatu koneksi namun *output* LLM salah mentafsirkan menjadi menghapus firewall atau mengizinkan suatu koneksi. Ini membuktikan bahwa model LLM prompt yang diinputkan harus lebih disensitif atau dioptimalkan untuk *intent* pengguna yang terkadang multi-tafsir.

2) *Kurangnya Pemahaman Chain Mikrotik*: *Test case* nomor IN-AC-01 dan IN-AC-05 kesalahan terdapat pada *chain*. Hal ini bisa terjadi karena disaat pengujian peneliti hanya memberikan informasi *chain* yang kurang detail. Ini membuktikan bahwa optimasi *prompt* harus dilakukan agar model LLM tahu apa yang sebenarnya dan detail mengenai suatu perintah.

D. *Pembahasan Rekomendasi Sistem*

Temuan-temuan anomali di atas menegaskan argumen bahwa penggunaan *base model* LLM (Qwen2.5-Coder-7B-Instruct) semata tidak cukup aman untuk diimplementasikan langsung pada *production environment* infrastruktur jaringan. Untuk menekan tingkat kegagalan tersebut, implementasi Retrieval-Augmented Generation (RAG) pada platform n8n terbukti sangat krusial untuk mencegah AI berhalusinasi mengenai data alamat IP.

Lebih jauh lagi, kegagalan AI dalam membedakan *chain* (input vs forward) dan kegagalan memahami negasi leksikal menunjukkan batas maksimal dari *Prompt Engineering*. Oleh karena itu, pendekatan Fine-Tuning model menjadi rekomendasi paling mendesak. Melalui *fine-tuning* menggunakan *dataset* khusus yang berisi ratusan pasang instruksi jaringan Bahasa Indonesia dan respons JSON MikroTik yang valid, model akan membangun pemahaman leksikal secara organik, sehingga tidak lagi bergantung pada *System Prompt* yang panjang dan mengurangi probabilitas kesalahan penentuan rantai (*chain*) serta tindakan (*action*).

Untuk mengatasi keterbatasan komputasi pada perangkat keras komoditas (*edge* atau *Small Office Home Office* / SOHO), pemanfaatan *Small Language Models* (SLM) pada rumpun parameter kecil (seperti varian 7B) menawarkan keseimbangan yang optimal antara efisiensi sumber daya dan performa ekstraksi instruksi [8]. Meskipun SLM memiliki konstrain fungsional saat dihadapkan pada perintah kompleks, kelemahan ini dapat dimitigasi secara komprehensif menggunakan *framework* validasi eksternal berbasis *JSON Schema* dan Pydantic [9]. Validasi non-LLM (*validator agent*) berperan sebagai lapisan pelindung (*guardrail*) yang memverifikasi keabsahan struktur dan keberadaan parameter spesifik (seperti tag yang dibutuhkan oleh *controller*), lalu merespons dengan umpan balik (*feedback*) otomatis agar model memperbaiki kesalahan *parsing* sebelum instruksi diterapkan pada jaringan [9], [10].

Seiring dengan pergeseran arsitektur menuju *Agentic-AI*, penyelesaian perintah majemuk (*multi-intent*) tidak lagi dibebankan pada satu model tunggal, melainkan didistribusikan ke dalam sistem *multi-agent* [11]. Melalui arsitektur ini, LLM dan agen non-LLM berkolaborasi dalam satu ekosistem: satu agen mendelegasikan dan menerjemahkan bahasa natural, sementara agen lain melakukan validasi konflik (*conflict detection*) dan otomatisasi *Infrastructure-as-Code* (IaC) secara tertutup (*closed-loop*) [10], [11]. Penerapan model agen kolaboratif telah terbukti mampu memberikan visibilitas terdistribusi secara *real-time* dan beroperasi dengan latensi minimal bahkan pada perangkat berarsitektur rendah (seperti implementasi R-Snort pada *Raspberry Pi* 5 di lingkungan SOHO) [12]. Kolaborasi multi-agen ini secara fundamental merevolusi operasional IBN yang kaku menjadi sistem yang sangat otonom, presisi, dan *user-friendly* [10], [11].

Sebagai penutup, keberhasilan sistem dalam mempertahankan rata-rata akurasi fungsional sebesar 90% pada pengujian dengan distribusi seimbang (sebagaimana dipetakan pada Tabel I) mengonfirmasi kelayakan operasional dari arsitektur integrasi yang diusulkan. Eksperimen berbasis variasi leksikal dan ragam bahasa nonformal yang dirinci pada Tabel II telah memberikan tolak ukur empiris mengenai sejauh mana model Qwen2.5-Coder-7B mampu melakukan penalaran semantik tanpa kehilangan konteks teknis. Meskipun teknik *Prompt Engineering* terbukti efektif menekan bias arah paket jaringan, sisa kegagalan translasi sebesar 10% kembali menegaskan batas ambang toleransi bahasa natural pada model berukuran

ISSN (Online) 2747-0563
Volume 6 Tahun 2026

kecil. Oleh karena itu, penerapan kerangka validasi eksternal berlapis (seperti *JSON Schema*) di sisi *runtime* platform n8n menjadi syarat mutlak untuk mencegah miskonfigurasi. Secara keseluruhan, perpaduan antara kemampuan abstraksi LLM dengan kedisiplinan eksekusi dari *workflow orchestration* ini berhasil meletakkan fondasi awal yang solid menuju implementasi jaringan SOHO yang sepenuhnya otonom, presisi, dan aman dari anomali sintaksis.

IV. KESIMPULAN DAN SARAN

Berdasarkan hasil pengujian dan analisis mendalam terhadap implementasi otomatisasi manajemen kebijakan *firewall* MikroTik menggunakan platform n8n dan *Large Language Model* (LLM) Qwen2.5-Coder-7B-Instruct, dapat disimpulkan bahwa arsitektur sistem yang diusulkan telah berhasil diimplementasikan dengan baik. Orkestrasi sistem yang menggabungkan *website* sebagai antarmuka *intent*, n8n sebagai pengelola alur kerja dan RAG, RunPod vLLM sebagai mesin inferensi HTTP, serta Supabase sebagai basis data, terbukti mampu memproses instruksi bahasa natural menjadi format keluaran JSON yang valid secara sintaksis. Model Qwen2.5-Coder-7B-Instruct menunjukkan kinerja yang cukup mumpuni dengan tingkat keberhasilan mencapai 90% dalam menerjemahkan maksud pengguna ke dalam parameter *firewall* secara terstruktur.

Meskipun demikian, sisa tingkat kegagalan sebesar 10% membuktikan hipotesis bahwa *base model* AI yang sangat bergantung pada pendekatan semantik masih memiliki kelemahan fundamental saat dihadapkan pada batasan leksikal yang kaku, seperti aturan baku pada RouterOS. Kelemahan ini termanifestasi dalam bentuk misinterpretasi arah paket lalu lintas, seperti tertukarnya penggunaan *chain input* dan *forward*, serta kegagalan dalam merespons instruksi negasi yang berakibat pada pemberian *action accept* untuk instruksi pemblokiran. Untuk mengatasi hal tersebut, penerapan teknik *Retrieval-Augmented Generation* (RAG) terbukti sangat krusial dalam menghilangkan halusinasi pembuatan alamat IP fiktif dengan cara menyuntikkan data topologi riil ke dalam instruksi. Selain itu, pengetatan *System Prompt* dengan aturan logis berfungsi sebagai mekanisme pertahanan agar AI mematuhi format yang disyaratkan oleh MikroTik, meskipun langkah ini belum sepenuhnya mampu menutupi celah misinterpretasi semantik.

Untuk meningkatkan keandalan, keamanan, dan Tingkat Kesiapterapan Teknologi (TKT) dari sistem otomatisasi jaringan ini di masa mendatang, terdapat beberapa rekomendasi strategis yang perlu dipertimbangkan. Pertama, sangat disarankan untuk tidak lagi hanya mengandalkan *base model* dan *System Prompt* yang membebani kinerja. Penelitian selanjutnya direkomendasikan untuk melakukan *fine-tuning* pada model Qwen menggunakan *dataset* spesifik yang berisi ribuan pasangan instruksi bahasa Indonesia dan JSON MikroTik yang valid. Langkah ini akan membentuk insting leksikal bawaan pada model, sehingga dapat secara signifikan meminimalisasi kesalahan logika *chain* dan *action*.

Selanjutnya, sebelum instruksi JSON diinjeksi secara otomatis ke perangkat *router* fisik, alur kerja di platform n8n harus dilengkapi dengan validasi berlapis atau *Pre-Flight Check*. Mekanisme validasi tambahan berupa *sandbox* atau *dry-run* ini berfungsi untuk memblokir eksekusi secara otomatis jika mendeteksi anomali berbahaya, seperti pemberian izin akses ke alamat IP *router* yang krusial atau perintah penghapusan yang dapat merusak stabilitas konfigurasi sistem. Terakhir, pada iterasi pengembangan berikutnya, variasi *test case* pengujian perlu diperluas agar tidak hanya mencakup rantai lalu lintas dasar, tetapi juga mencakup skenario manajemen lalu lintas lanjutan seperti *Quality of Service* (QoS) atau kebijakan *routing* dinamis guna menguji daya tahan logika model secara lebih komprehensif.

UCAPAN TERIMA KASIH

Penulis mengucapkan rasa syukur dan terima kasih yang sebesar-besarnya kepada Bapak Achmad Junaidi, S.Kom, M.Kom dan Ibu Eva Yulia Puspaningrum, S.Kom, M.Kom, selaku dosen pembimbing di Program Studi Informatika, Universitas Pembangunan Nasional "Veteran" Jawa Timur. Segala arahan, bimbingan, serta diskusi teknis yang solutif mengenai topologi jaringan dan arsitektur kecerdasan buatan sangatlah bernilai selama proses penelitian dan perancangan sistem otomatisasi ini berlangsung. Terakhir, penghargaan dan terima kasih disampaikan kepada Tim SANTIKA yang telah meluangkan waktu untuk membuat dan menyediakan panduan serta *template* pemformatan dokumen ini, sehingga penulisan laporan penelitian dapat tersusun dengan rapi dan sesuai standar publikasi.

REFERENSI

[1] M. W. S. Utomo, H. E. Wahanani, dan A. Junaidi, "Validasi keamanan infrastruktur web: analisis jaringan dan kekuatan enkripsi SSL/TLS instansi A," *Prosiding Seminar Nasional Penelitian dan Pengabdian Kepada Masyarakat LPPM Universitas 'Aisyiyah Yogyakarta*, vol. 4, hlm. 696–705, Mar 2026, [Daring]. Tersedia pada: https://proceeding.unisayogya.ac.id/index.php/prosemnaslppm/article/view/2207

[2] D. P. Lasminingrum, E. Y. Puspaningrum, dan B. M. Mulyo, "Perbandingan metode lexical dan semantic retrieval : BM25, IndoSBERT baseline, dan IndoSBERT fine-tuned pada pencarian dokumen berbahasa Indonesia," *Prosiding Seminar Nasional Penelitian dan Pengabdian Kepada Masyarakat LPPM Universitas 'Aisyiyah Yogyakarta*, vol. 4, Mar 2026, [Daring]. Tersedia pada: https://proceeding.unisayogya.ac.id/index.php/prosemnaslppm/article/view/2088

[3] F. Liu, B. Farkiani, dan P. Crowley, "Large Language Models for computer networking operations and management: A survey on applications, key techniques, and opportunities," 1 Oktober 2025, *Elsevier B.V.* doi: 10.1016/j.comnet.2025.111614.

[4] S. Cruzes, "Revolutionizing optical networks: The integration and impact of large language models," 1 Oktober 2025, *Elsevier B.V.* doi: 10.1016/j.osn.2025.100812.

[5] D. Munson, P. Alain, dan G. Doyen, "NEAT: A Nile-English Aligned Translation corpus based on a robust methodology for Intent Based Networking," *Computer Networks*, vol. 271, Okt 2025, doi: 10.1016/j.comnet.2025.111519.

[6] Mr. S. Pawar, "A Practical Evaluation of Self-Hosted n8n for Secure and Scalable Workflow Automation," *INTERNATIONAL JOURNAL*

OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT, vol. 09, no. 06, hlm. 1–9, Jun 2025, doi: 10.55041/ijsrem50302.
[7] A. B. Pratomo, "Automation of Cyber Attack Mitigation on RouterOS Using n8n Orchestration Platform," *Bulletin of Network Engineer and Informatics*, vol. 4, no. 1, hlm. 1–5, Apr 2026, doi: 10.59688/fq83hq56.
[8] M. Mahade Hasan, M. Waseem, K. K. Kemell, J. Rasku, J. Ala-Rantala, dan P. Abrahamsson, "Assessing small language models for code generation: An empirical study with benchmarks," *Journal of Systems and Software*, vol. 236, Jun 2026, doi: 10.1016/j.jss.2026.112815.
[9] D. Rohrschneider, M. Pehlke, U. Handmann, dan M. Jansen, "LLM-based JSON Mapping and Blockchain Integration for Digital Product Passports," *Digital Business*, vol. 6, no. 1, hlm. 100167, Jun 2026, doi: 10.1016/j.digbus.2026.100167.
[10] M. K. Hossain dan W. Aljoby, "NetIntent: Leveraging Large Language Models for End-to-End Intent-Based SDN Automation," *IEEE Open Journal of the Communications Society*, vol. 6, hlm. 10512–10541, 2025, doi: 10.1109/OJCOMS.2025.3642642.
[11] D. Brodimas, A. Birbas, D. Kapolos, dan S. Denazis, "Intent-Based Infrastructure and Service Orchestration Using Agentic-AI," *IEEE Open Journal of the Communications Society*, vol. 6, hlm. 7150–7168, 2025, doi: 10.1109/OJCOMS.2025.3600706.
[12] J. G. López, D. O. P. Tabacu, N. P. Soriano, dan A. A. García, "R-Snort: A Performance-Optimized Multi-Agent NIDS Architecture for SOHO and Edge-of-Things Networks Using Snort 3 on Raspberry Pi 5," *Computers*, vol. 15, no. 5, Mei 2026, doi: 10.3390/computers15050270.