Evaluasi Efektivitas Wazuh SIEM terhadap Ancaman Injeksi REST API Menggunakan Metode Eksperimental

Evaluation of Wazuh SIEM Effectiveness Against REST API Injection Threats Using an Experimental Method

Salsabila Hanafiah[1], Muhammad Idris[2]

Program Studi Rekayasa Keamanan Siber, Politeknik Negeri Batam, Batam, Indonesia
E-mail: salsabilahanafiah71@gmail.com

Informasi Artikel
Naskah diterima 23 Jun 2026
Direvisi 23 Jul 2026
Dipublikasikan 23 Ags 2026

Kata kunci:
SIEM; REST API; Injeksi; Wazuh; Keamanan Siber.

Abstrak

Security Information and Event Management (SIEM) merupakan solusi kunci dalam mendeteksi dan merespons ancaman keamanan siber secara real-time. Penelitian ini mengevaluasi efektivitas Wazuh, sebuah platform SIEM open source berbasis deteksi intrusi host (Host-based Intrusion Detection System/HIDS), dalam mendeteksi serangan injeksi pada REST API, meliputi SQL Injection, Command Injection, dan LDAP Injection. Pendekatan eksperimental digunakan dengan mensimulasikan serangan pada REST API yang sengaja dibuat rentan (DVWS-Node). Parameter yang dianalisis mencakup tingkat keberhasilan deteksi, waktu deteksi, dan kelengkapan informasi alert. Hasil penelitian menunjukkan Wazuh mampu mendeteksi 100% serangan yang diujikan dengan rata-rata waktu deteksi 1,57 detik, serta menampilkan seluruh lima komponen informasi yang dibutuhkan untuk analisis forensik, yaitu source IP, endpoint target, jenis serangan, severity, dan payload indicator. Hasil ini menunjukkan bahwa Wazuh secara mandiri layak digunakan sebagai solusi SIEM untuk mendeteksi ancaman injeksi pada REST API di lingkungan open source.

Article Info
Received Jun 23, 2026
Revised Jul 23, 2025
Published Aug 23, 2026

Keyword:
SIEM; REST API; Injection; Wazuh; Cybersecurity.

Abstract

Security Information and Event Management (SIEM) serves as a critical solution for detecting and responding to cybersecurity threats in real-time. This study evaluates the effectiveness of Wazuh, an open-source SIEM platform based on Host-based Intrusion Detection System (HIDS), in detecting injection attacks targeting REST APIs, including SQL Injection, Command Injection, and LDAP Injection. An experimental approach was employed by simulating attacks on a purposely vulnerable REST API server (DVWS-Node). Parameters analyzed include detection success rate, detection time, and alert information completeness. Results indicate that Wazuh achieved 100% attack detection with an average detection time of 1.57 seconds, and displayed all five information components required for forensic analysis: source IP, target endpoint, attack type, severity, and payload indicator. These results show that Wazuh, on its own, is a viable SIEM solution for detecting injection threats on REST APIs in an open-source environment.

Corresponding Author:
Muhammad Idris, Program Studi Rekayasa Keamanan Siber, Politeknik Negeri Batam
Email: Idris@polibatam.ac.id

1. Pendahuluan

Pengembangan sistem informasi memiliki potensi untuk meningkatkan mutu dan kualitas suatu organisasi. Pentingnya nilai informasi mengharuskan akses terhadap informasi yang dihasilkan oleh sistem harus dibatasi hanya pada individu tertentu saja, agar integritas informasi yang disampaikan tetap terjaga. Kebocoran informasi kepada pihak yang tidak berwenang dapat mengakibatkan kerugian bagi organisasi [1].

Keamanan informasi merupakan aspek yang sangat penting bagi setiap instansi untuk terlindung dari gangguan atau tindakan kriminal. Menurut laporan dari internet traffic watchdog, pada tahun 2016, sekitar 90% kejahatan dunia maya terjadi melalui serangan pada aplikasi web, dengan metode yang paling banyak adalah injeksi pada database, yakni mencapai 47,06% dari total serangan yang tercatat [1].

Security Information and Event Management (SIEM) berfungsi untuk mengumpulkan data log dari berbagai komponen infrastruktur teknologi informasi, termasuk server, perangkat jaringan, aplikasi, dan perangkat keamanan seperti firewall dan antivirus. Data yang terkumpul kemudian dianalisis menggunakan algoritma canggih untuk mengidentifikasi pola atau aktivitas yang mencurigakan [2]. Konsep SIEM ini pertama kali diperkenalkan oleh Gartner pada tahun 2005 sebagai hasil penggabungan Security Event Management (SEM) dan Security Information Management (SIM) yang bekerja dengan cara mengumpulkan log, menormalisasi, agregasi, serta melakukan korelasi untuk menghasilkan visualisasi dan peringatan [7].

Dalam perkembangannya, implementasi SIEM telah banyak diteliti oleh para ahli. Menurut penelitian yang dilakukan oleh Al Amin dkk. [2], korelasi data log dari berbagai sumber melalui arsitektur SIEM terbukti

krusial untuk menghadapi ancaman secara real-time sekaligus memastikan kepatuhan regulasi. Lebih lanjut, dalam kajian yang dipaparkan oleh Hadi dkk. [4], penggunaan kombinasi platform SIEM seperti Wazuh, Suricata, dan Telegram terbukti efektif untuk melakukan deteksi serta analisa insiden keamanan secara spesifik pada perangkat web server. Di sisi lain, Sijabat dkk. [6] juga memaparkan perancangan sistem SIEM berbasis Elastic Stack yang berhasil mendeteksi berbagai simulasi serangan yang masuk dalam cakupan kategori OWASP Top 10. Selain aspek deteksi melalui SIEM, Prasetyo dkk. [5] menjelaskan bahwa strategi pengamanan website dari serangan siber seperti DoS, SQL Injection, dan XSS akan menjadi jauh lebih optimal jika dikombinasikan dengan penggunaan Web Application Firewall (WAF).

Serangan injeksi sendiri merupakan salah satu kerentanan yang terjadi akibat input pengguna yang tidak divalidasi, dan merupakan metode eksploitasi yang paling umum digunakan dalam konteks aplikasi web. Sebagai contoh, SQL Injection merupakan teknik yang sangat sering digunakan oleh penyerang untuk berkomunikasi secara ilegal dengan basis data yang terhubung ke aplikasi, mengambil informasi sensitif, dan mengendalikan aplikasi atau sistem [3]. Contoh teknik serangan ini bekerja dengan memanipulasi logika perintah SQL guna memperoleh akses ke database dan menjadi salah satu ancaman yang paling krusial dalam jaringan [3].

Untuk memitigasi risiko dari berbagai serangan siber tersebut, diperlukan solusi monitoring keamanan yang mampu bekerja secara real-time dan menyeluruh. Salah satu solusi yang banyak digunakan adalah platform Wazuh, yang berfungsi sebagai sistem deteksi intrusi berbasis host (Host-based Intrusion Detection System/HIDS) dengan kapabilitas analisis log, pemeriksaan integritas berkas, pemantauan registri, deteksi rootkit, hingga eksekusi respons aktif [8]. Wazuh dipilih sebagai solusi tunggal dalam penelitian ini karena memiliki mesin korelasi log (*ruleset*) yang dapat dikustomisasi secara spesifik untuk mengenali pola serangan pada trafik HTTP, termasuk serangan injeksi yang menyasar REST API. Selain itu, arsitektur Wazuh yang ringan (*lightweight*) namun kuat menjadikannya sesuai untuk dipasang sebagai agen pemantauan pada web server (Nginx/Apache) yang meng-hosting REST API tanpa membebani kinerja server secara signifikan. Kapabilitas Wazuh dalam melakukan *decoding*, klasifikasi, dan pemberian level *severity* secara otomatis terhadap log yang masuk juga memungkinkan proses analisis forensik yang lebih cepat dan terstruktur dibandingkan pemrosesan log secara manual.

Berdasarkan latar belakang tersebut, penelitian ini berfokus untuk menguji efektivitas platform Wazuh dalam mendeteksi serta merespons serangan injeksi pada REST API. Selain itu, studi ini juga diarahkan untuk mengidentifikasi berbagai kerentanan injeksi melalui pendekatan eksperimental, yang pada akhirnya hasil analisis tersebut dapat digunakan sebagai landasan rekomendasi untuk meningkatkan strategi pertahanan keamanan instansi secara menyeluruh.

2. Metode Penelitian

Penelitian ini menerapkan pendekatan eksperimental guna mengevaluasi secara mendalam kinerja platform Wazuh dalam mendeteksi serangan injeksi terhadap REST API. Pendekatan eksperimental dipilih karena memberikan keleluasaan bagi peneliti untuk menciptakan lingkungan uji yang terkendali, di mana skenario serangan dapat disimulasikan secara langsung, dengan mempertimbangkan indikator kunci: kemampuan mendeteksi serangan injeksi, tingkat kelengkapan dan akurasi log, kualitas informasi dalam peringatan (*alert*), serta kecepatan respon sistem.

2.1. Justifikasi Parameter Penelitian

Pemilihan tiga parameter utama, yaitu tingkat keberhasilan deteksi, waktu deteksi, dan kelengkapan informasi *alert*, didasarkan pada kebutuhan operasional Security Operations Center (SOC) dalam menilai kelayakan suatu solusi SIEM secara menyeluruh. Tingkat keberhasilan deteksi dipilih karena merepresentasikan kemampuan dasar suatu SIEM dalam mengenali ancaman; parameter ini menjadi prasyarat mutlak, sebab SIEM yang tidak mampu mendeteksi serangan tidak akan memberikan nilai keamanan sama sekali [2]. Waktu deteksi dipilih sebagai parameter kedua karena kecepatan respons berkaitan langsung dengan mitigasi risiko; semakin cepat suatu serangan terdeteksi, semakin kecil peluang penyerang untuk mengeksploitasi celah keamanan lebih jauh [7]. Kelengkapan informasi *alert* dipilih sebagai parameter ketiga karena keberhasilan deteksi maupun kecepatan respons tidak akan bermanfaat secara operasional apabila informasi yang dihasilkan tidak cukup untuk mendukung analisis forensik; kelima komponen yang dievaluasi, yaitu source IP, endpoint target, jenis serangan, severity, dan payload indicator, merupakan elemen minimal yang lazim dibutuhkan analis keamanan untuk melakukan investigasi insiden secara cepat dan akurat [4].

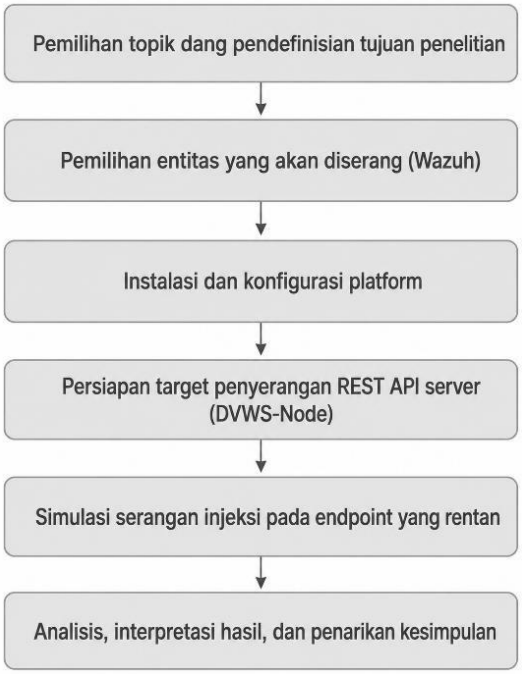
Kendali terhadap variabel penelitian diwujudkan melalui sejumlah perlakuan tetap (*fixed treatment*) yang diterapkan secara konsisten sepanjang pengujian. Pertama, topologi jaringan dijaga tetap berupa dua virtual machine yang saling terhubung dalam satu jaringan lokal tanpa perubahan konfigurasi selama proses pengujian berlangsung. Kedua, tidak terdapat trafik latar belakang (*background traffic*) lain pada segmen jaringan yang sama, sehingga seluruh log yang tercatat oleh Wazuh murni berasal dari payload pengujian yang dikirimkan peneliti. Ketiga, format payload distandardisasi untuk masing-masing jenis serangan, yaitu `admin' --` untuk SQL Injection, `192.168.1.32; vhoami` untuk Command Injection, dan `*)(I(8` untuk LDAP Injection. Keempat, mekanisme pencatatan waktu dilakukan sepenuhnya secara otomatis oleh sistem Wazuh, mulai dari tahap *log ingestion* hingga status *alert triggered* pada dashboard, sehingga menghilangkan potensi bias akibat pencatatan waktu secara manual.

2.2. Metode Pengumpulan Data

Dalam penelitian ini, teknik pengumpulan data menggunakan metode eksperimental dengan memanfaatkan data kuantitatif dari koleksi log untuk mengukur kecepatan deteksi ancaman. Pemantauan dilakukan dengan mengukur waktu respons Wazuh terhadap serangan simulasi, yang seluruhnya dicatat secara otomatis oleh sistem.

2.3. Prosedur Penelitian

Prosedur penelitian dilaksanakan melalui tahapan-tahapan yang terstruktur sebagaimana ditunjukkan pada Gambar 1: (1) pemilihan topik dan pendefinisian tujuan penelitian; (2) instalasi dan konfigurasi Wazuh-Server serta penempatan Wazuh-Agent pada target; (3) persiapan target penyerangan REST API server (DVWS-Node); (4) simulasi serangan injeksi pada endpoint yang rentan; dan (5) analisis, interpretasi hasil, serta penarikan kesimpulan.



Gambar 1. Prosedur Penelitian

Tahapan prosedur dalam penelitian ini dilaksanakan secara sistematis yang diawali dengan pemilihan topik serta pendefinisian tujuan penelitian guna menetapkan arah dan batasan kajian yang jelas. Langkah berikutnya adalah proses instalasi serta konfigurasi teknis pada Wazuh-Server, yang

kemudian dilanjutkan dengan penempatan Wazuh-Agent pada target REST API agar seluruh aktivitas pada endpoint dapat dipantau secara real-time. Secara paralel, dipersiapkan pula server REST API yang rentan yaitu Damn Vulnerable Web Services (DVWS-Node) sebagai target penyerangan utama. Tahap krusial berikutnya adalah melakukan simulasi serangan injeksi secara langsung pada endpoint target yang telah teridentifikasi memiliki celah keamanan. Rangkaian prosedur ini kemudian diakhiri dengan analisis mendalam terhadap performa deteksi Wazuh, interpretasi komprehensif atas hasil data log yang diperoleh, serta penarikan kesimpulan sebagai capaian akhir dari penelitian.

2.4. Arsitektur Pengumpulan Log

Arsitektur pengumpulan log pada penelitian ini dirancang agar seluruh aktivitas pada REST API dapat terekam dan dianalisis secara real-time oleh Wazuh. Wazuh-Agent yang terpasang pada VM target REST API (DVWS-Node) membaca berkas log web server, yaitu `/var/log/nginx/access.log`, secara berkala. Setiap entri log yang baru kemudian dikirimkan secara aman melalui port 1514 menuju Wazuh-Manager yang berjalan pada VM terpisah (IP 192.168.74.133). Pada sisi Wazuh-Manager, log yang diterima diproses melalui tahapan *decoding* dan pencocokan terhadap *ruleset* yang telah dikonfigurasi khusus untuk mengenali pola-pola payload injeksi pada HTTP POST Body, sebelum akhirnya dimunculkan sebagai *alert* pada dashboard Security Events.

2.5. Metode Analisis Data

Metode analisis data difokuskan pada evaluasi kemampuan deteksi serangan injeksi REST API oleh Wazuh. Aspek yang dianalisis meliputi kemunculan *alert*, tingkat sensitivitas deteksi, kelengkapan informasi log, serta kejelasan dan struktur informasi dalam *alert* termasuk klasifikasi ancaman dan tingkat keparahan (*severity*).

3. Hasil dan Pembahasan

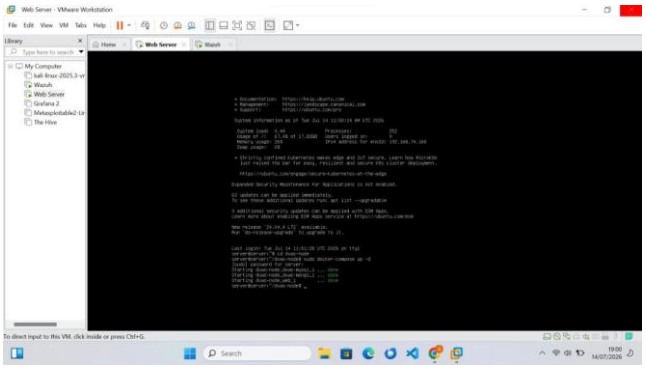
Pengujian dilakukan untuk mengevaluasi kemampuan platform Wazuh dalam mendeteksi serangan *injection* pada REST API. Tiga jenis serangan yang digunakan meliputi SQL Injection, Command Injection, dan LDAP Injection. Parameter yang dianalisis mencakup tingkat keberhasilan deteksi, waktu deteksi, serta kelengkapan informasi *alert*.

3.1. Implementasi dan Konfigurasi

Implementasi sistem dilakukan pada dua virtual machine (VM) yang saling terhubung dalam satu jaringan lokal, yaitu VM Wazuh sebagai server deteksi dan VM DVWS-Node sebagai target REST API yang rentan. Sebelum instalasi komponen SIEM, seluruh virtual machine dikonfigurasi dan diverifikasi terlebih dahulu, dengan VM Wazuh menggunakan IP 192.168.74.133 yang berhasil diakses dan siap digunakan sebagai server Wazuh.

Wazuh diimplementasikan sebagai sistem deteksi intrusi berbasis host pada VM dengan IP 192.168.74.133. Instalasi dilakukan langsung pada sistem operasi Ubuntu Server tanpa kontainerisasi. Setelah instalasi selesai, Wazuh-Server dikonfigurasi untuk menerima log dari Wazuh-Agent yang dipasang pada target REST API, sehingga seluruh aktivitas serangan pada endpoint dapat dipantau secara real-time melalui dashboard Wazuh.

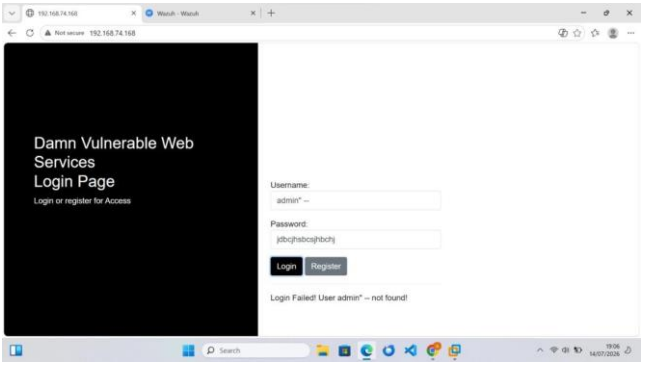
Web aplikasi rentan yang digunakan sebagai objek uji injeksi adalah DVWS-Node (Damn Vulnerable Web Services Node) pada VM dengan IP 192.168.74.168. DVWS-Node merupakan aplikasi web yang sengaja dirancang memiliki celah keamanan untuk keperluan pengujian. Aplikasi ini dikonfigurasi menggunakan Docker Compose yang menjalankan tiga kontainer secara bersamaan, yaitu *dvws-mongo* (database MongoDB), *dvws-mysql* (database MySQL), dan *web* (aplikasi REST API rentan). Gambar 2 menunjukkan ketiga kontainer berhasil dijalankan dengan status *done*, menandakan web aplikasi rentan siap digunakan sebagai objek simulasi serangan injeksi.



Gambar 2. Implementasi Web Aplikasi Rentan DVWS-Node sebagai Objek Uji Injeksi pada VM IP 192.168.74.168

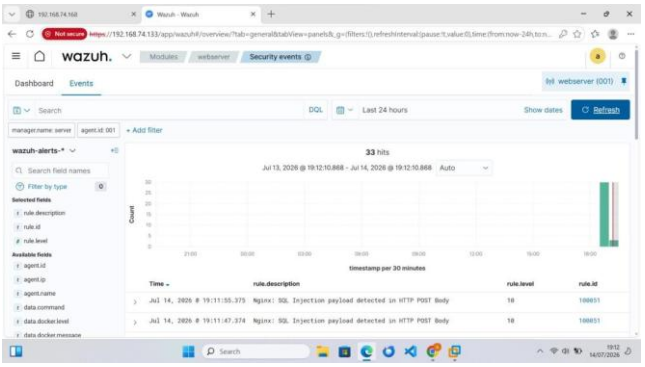
3.2. Pengujian SQL Injection

Setelah seluruh komponen berhasil diimplementasikan, pengujian SQL Injection dilakukan dengan mengakses halaman login web aplikasi rentan DVWS-Node melalui browser. Payload SQL Injection dimasukkan pada kolom username dengan format `admin' --` yang merupakan teknik bypass autentikasi klasik, sementara kolom password diisi dengan nilai sembarang. Gambar 3 menunjukkan tampilan halaman login DVWS-Node saat payload injeksi dimasukkan sebelum tombol Login ditekan.



Gambar 3. Tampilan Web Rentan DVWS-Node saat Payload SQL Injection Dimasukkan pada Kolom Username

Setelah payload dikirimkan, Wazuh secara real-time mendeteksi adanya aktivitas mencurigakan pada web-server. Gambar 4 menunjukkan dashboard Wazuh pada tab Security Events yang berhasil mencatat serangkaian *alert* SQL Injection dengan deskripsi *Nginx: SQL Injection payload detected in HTTP POST Body* pada rule ID 100051 dengan level severity 10, menandakan seluruh pengiriman payload berhasil terdeteksi secara otomatis dan real-time oleh Wazuh.



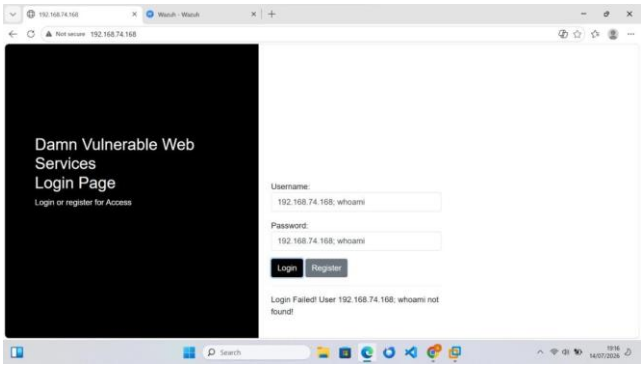
Gambar 4. Dashboard Wazuh Menampilkan Alert SQL Injection dengan Rule ID 100051 Level 10

Proses deteksi ini diawali dengan tahap *decoding*, di mana Wazuh-Manager mengekstrak isi HTTP POST Body dari log akses Nginx menggunakan

decoder yang dikonfigurasi untuk mengenali format request pada endpoint login DVWS-Node. Isi payload yang telah diekstrak kemudian dicocokkan terhadap *ruleset* yang memuat pola string dan karakter khusus (*meta-characters*) yang umum digunakan dalam teknik SQL Injection, seperti tanda kutip tunggal dan penanda komentar SQL (’ −). Ketika pola tersebut cocok dengan payload yang terekam, Wazuh memicu rule dengan ID 100051 dan level severity 10.

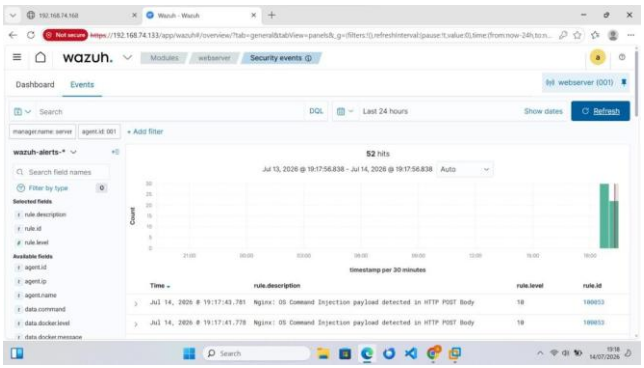
3.3. Pengujian OS Command Injection

Pengujian OS Command Injection dilakukan dengan mengakses halaman login web aplikasi rentan DVWS-Node melalui browser. Payload OS Command Injection dimasukkan pada kolom username dan password dengan format `192. 168. 1. 32; whoami` yang merupakan teknik injeksi perintah sistem operasi untuk mengeksekusi perintah `whoami` pada server.



Gambar 5. Tampilan Web Rentan DVWS-Node saat Payload OS Command Injection Dimasukkan pada Kolom Username dan Password

Setelah payload dikirimkan, Wazuh secara real-time mendeteksi adanya aktivitas mencurigakan pada web-server. Dashboard Wazuh pada tab Security Events berhasil mencatat serangkaian *alert* OS Command Injection dengan deskripsi *Nginx: OS Command Injection payload detected in HTTP POST Body* pada rule ID 100053 dengan level severity 10.

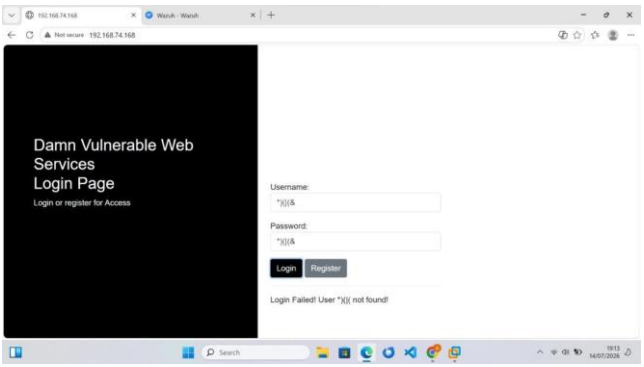


Gambar 6. Dashboard Wazuh Menampilkan Alert OS Command Injection dengan Rule ID 100053 Level 10

Pada pengujian ini, *ruleset* Wazuh mengenali payload berdasarkan keberadaan karakter pemisah perintah shell (`;`) yang diikuti oleh perintah sistem operasi (`vhoami`) pada isi HTTP POST Body. Kecocokan pola tersebut memicu rule dengan ID 100053 dan level severity 10, yang mengindikasikan bahwa aktivitas ini berpotensi memberikan akses eksekusi perintah kepada penyerang.

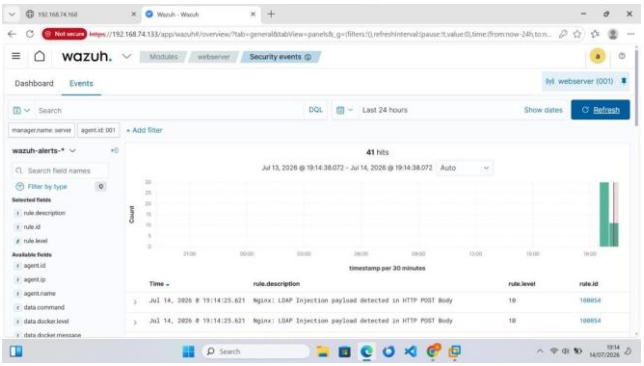
3.4. Pengujian LDAP Injection

Pengujian LDAP Injection dilakukan dengan mengakses halaman login web aplikasi rentan DVWS-Node melalui browser. Payload LDAP Injection dimasukkan pada kolom username dan password dengan format `*) ( I ( 8` yang merupakan teknik manipulasi filter LDAP untuk membypass mekanisme autentikasi.



Gambar 7. Tampilan Web Rentan DVWS-Node saat Payload LDAP Injection Dimasukkan pada Kolom Username dan Password

Setelah payload dikirimkan, Wazuh secara real-time mendeteksi adanya aktivitas mencurigakan pada web-server. Dashboard Wazuh pada tab Security Events berhasil mencatat serangkaian *alert* LDAP Injection dengan deskripsi *Nginx: LDAP Injection payload detected in HTTP POST Body* pada rule ID 100054 dengan level severity 10.



Gambar 8. Dashboard Wazuh Menampilkan Alert LDAP Injection dengan Rule ID 100054 Level 10

Pada pengujian ini, *ruleset* Wazuh mengenali payload berdasarkan keberadaan operator logika dan karakter filter LDAP (`*, I , 8`) pada isi HTTP POST Body yang mengindikasikan upaya manipulasi filter pencarian LDAP. Kecocokan pola tersebut memicu rule dengan ID 100054 dan level severity 10.

3.5. Deteksi Serangan

Berdasarkan hasil pengujian, Wazuh memiliki tingkat keberhasilan deteksi yang sama terhadap seluruh serangan. Total 30/30 atau 100% serangan dapat terdeteksi oleh Wazuh, sebagaimana ditunjukkan pada Tabel 1.

Tabel 1. Tingkat Keberhasilan Deteksi Serangan

Jenis Serangan	Wazuh
SQL Injection	10/10
Command Injection	10/10
LDAP Injection	10/10
Total	**30/30**

3.6. Waktu Deteksi

Waktu deteksi dihitung dari saat payload dikirimkan hingga *alert* muncul pada dashboard Wazuh. Seluruh perhitungan waktu dilakukan secara otomatis oleh sistem, mulai dari *log ingestion* (penerimaan log dari Wazuh-Agent), proses *decoding* dan pencocokan *ruleset*, hingga status *alert triggered* pada dashboard, sehingga tidak terdapat intervensi pencatatan waktu secara manual yang dapat memengaruhi akurasi pengukuran. Tabel 2 menyajikan rata-rata waktu deteksi untuk setiap jenis serangan.

Tabel 2. Rata-rata Waktu Deteksi Serangan

Jenis Serangan	Wazuh (detik)
SQL Injection	1,5
Command Injection	1,7
LDAP Injection	1,5
Rata-rata	**1,57**

Wazuh memiliki rata-rata waktu deteksi 1,57 detik untuk keseluruhan jenis serangan yang diujikan, dengan waktu deteksi tercepat pada skenario SQL Injection dan LDAP Injection (1,5 detik), serta waktu deteksi sedikit lebih lama pada skenario Command Injection (1,7 detik).

3.7. Kelengkapan Informasi Alert

Kelengkapan *alert* dievaluasi berdasarkan lima komponen utama yang dibutuhkan untuk analisis forensik, sebagaimana ditunjukkan pada Tabel 3 beserta contoh konkret nilai yang berhasil ditangkap oleh Wazuh.

Tabel 3. Kelengkapan Informasi Alert pada Wazuh

Komponen Forensik	Status	Contoh Nilai
Source IP	✓	192.168.74.168
Endpoint Target	✓	/login
Jenis Serangan	✓	SQLi/CMDi/LDAPi
Severity	✓	Level 10
Payload Indicator	✓	admin' – ; whoami ; *)(|(&

Wazuh berhasil menampilkan seluruh lima komponen informasi *alert* secara lengkap, mulai dari source IP dan endpoint target yang menjadi sasaran serangan, klasifikasi jenis serangan, tingkat keparahan (*severity*), hingga indikator payload yang terdeteksi. Kelengkapan informasi ini secara langsung mendukung proses investigasi insiden karena analis dapat segera mengidentifikasi asal, target, jenis, tingkat urgensi, dan bukti konkret dari suatu insiden tanpa perlu menelusuri log mentah secara manual.

3.8. Analisis Per Jenis Serangan

Berdasarkan hasil pengujian terhadap klaster serangan SQL Injection, Wazuh menunjukkan kapabilitas pertahanan yang solid dengan tingkat keberhasilan deteksi mutlak mencapai 100% (10 dari 10 parameter uji berhasil diidentifikasi) dan rata-rata waktu deteksi 1,5 detik. Efisiensi ini didukung oleh subsistem kalkulasi dan penandaan waktu (*timestamping*) yang berjalan secara otomatis pada backend Wazuh-Manager, sejak proses *log ingestion* hingga *alert* dimunculkan.

Pada implementasi skenario pengujian Command Injection, Wazuh kembali menunjukkan performa deteksi yang konsisten dengan berhasil mengidentifikasi seluruh rangkaian serangan tanpa adanya *false negative* (tingkat keberhasilan 10/10), dengan rata-rata waktu deteksi 1,7 detik. Waktu deteksi yang sedikit lebih lama dibandingkan SQL Injection ini mengindikasikan bahwa proses pencocokan pola pada payload Command Injection, yang melibatkan pengenalan karakter pemisah perintah shell (;) beserta perintah sistem operasi yang menyertainya, memerlukan tahapan pemrosesan yang sedikit lebih kompleks pada *ruleset* Wazuh.

Evaluasi empiris pada klaster serangan LDAP Injection memperlihatkan bahwa Wazuh mampu mempertahankan akurasi deteksi yang sempurna dengan tingkat keberhasilan pemantauan menyentuh angka 100% (10/10), dengan rata-rata waktu deteksi 1,5 detik yang konsisten dengan performa deteksi pada skenario SQL Injection. Hal ini mengindikasikan bahwa *ruleset* Wazuh secara konsisten mampu mengenali pola-pola karakter khusus pada berbagai jenis serangan injeksi tanpa perbedaan performa yang signifikan.

3.9. Pembahasan

Berdasarkan hasil pengujian, Wazuh dievaluasi melalui tiga kriteria utama: kemampuan mendeteksi serangan, kecepatan memberikan peringatan, dan kelengkapan informasi yang disediakan bagi analis keamanan. Ditinjau dari kemampuan mendeteksi serangan, Wazuh menunjukkan performa yang sempurna dengan tingkat keberhasilan mencapai 100% pada seluruh jenis serangan yang diujikan, membuktikan bahwa *ruleset* yang dikonfigurasi mampu mengenali pola-pola payload injeksi secara konsisten pada trafik HTTP.

Dari sisi kecepatan respons, Wazuh mencatatkan rata-rata waktu deteksi 1,57 detik yang murni merepresentasikan proses internal Wazuh-Manager, mulai dari *log ingestion*, *decoding*, hingga pencocokan *ruleset*, tanpa adanya intervensi pencatatan waktu secara manual. Pengendalian variabel tersebut kemudian tercermin pada hasil pengujian, di mana rata-rata waktu deteksi antar jenis serangan hanya menunjukkan variasi yang sangat kecil, yaitu berkisar antara 1,5 hingga 1,7 detik. Konsistensi rentang waktu deteksi ini menjadi bukti empiris bahwa lingkungan pengujian berhasil dikendalikan secara ketat, sehingga selisih waktu deteksi yang teramati dapat diatribusikan secara valid kepada perbedaan kompleksitas pencocokan pola pada *ruleset* Wazuh untuk masing-masing jenis payload, bukan disebabkan oleh gangguan variabel eksternal yang tidak terkendali seperti beban jaringan atau trafik lain di luar skenario pengujian.

Dari sisi kelengkapan informasi peringatan, Wazuh berhasil menampilkan seluruh 5 komponen informasi forensik yang dibutuhkan, yaitu source IP, endpoint target, jenis serangan, severity, dan payload indicator. Secara keseluruhan, Wazuh terbukti sebagai solusi SIEM yang layak digunakan untuk mendeteksi serangan injeksi REST API karena menggabungkan akurasi deteksi, kecepatan respons otomatis, dan kelengkapan informasi forensik dalam satu platform yang ringan dan relatif mudah diimplementasikan.

4. Kesimpulan

Berdasarkan hasil evaluasi terhadap platform Wazuh sebagai solusi SIEM open source dalam mendeteksi serangan injeksi REST API, dapat disimpulkan bahwa Wazuh mampu mendeteksi 100% serangan yang disimulasikan, mencakup SQL Injection, Command Injection, dan LDAP Injection, dengan rata-rata waktu deteksi 1,57 detik yang diproses secara otomatis. Wazuh juga menampilkan seluruh lima komponen informasi yang diperlukan untuk analisis forensik, yaitu source IP, endpoint target, jenis serangan, severity, dan payload indicator. Dengan demikian, Wazuh merupakan pilihan yang layak digunakan sebagai platform SIEM mandiri untuk mendeteksi serangan injeksi pada REST API di lingkungan open source.

4.1. Saran

Penelitian ini masih memiliki sejumlah keterbatasan yang perlu ditindaklanjuti pada penelitian selanjutnya. Pertama, setiap jenis serangan pada penelitian ini hanya diuji dengan satu bentuk payload (misalnya SQL Injection hanya menggunakan format `admin' –`), sehingga belum mencakup variasi teknik lain seperti *union-based* maupun *blind/time-based* SQL Injection. Penelitian lanjutan disarankan untuk menguji variasi payload yang lebih luas guna memperkuat validitas generalisasi hasil efektivitas deteksi.

Kedua, penelitian ini belum melakukan pengujian terhadap trafik normal untuk mengukur tingkat *false positive* Wazuh, sehingga disarankan pengujian lanjutan turut menyertakan skenario tersebut agar tingkat akurasi deteksi dapat dievaluasi secara lebih menyeluruh.

Ketiga, penelitian selanjutnya dapat mengeksplorasi integrasi Wazuh dengan platform SIEM atau sistem respons insiden lain yang benar-benar independen sebagai mesin deteksi, sehingga apabila dilakukan perbandingan antarplatform pada penelitian mendatang, perbandingan tersebut dapat dilakukan secara lebih valid dan setara secara metodologis.

Keempat, untuk meningkatkan aspek visualisasi data, penelitian selanjutnya dapat menambahkan Grafana yang diintegrasikan dengan Wazuh guna menampilkan dashboard monitoring yang lebih interaktif dan mudah dipahami. Selain itu, untuk mendukung proses *incident response*, integrasi Wazuh dengan TheHive juga dapat dipertimbangkan agar penanganan insiden keamanan dapat dilakukan secara lebih terstruktur dan terkoordinasi.

Daftar Pustaka

[1] X. Doe, “Keamanan Sistem Informasi dan Ancaman Siber,” *Jurnal Keamanan Informasi*, vol. 5, no. 2, pp. 10–18, 2022.

[2] Muh. Al Amin et al., “Implementasi Security Information and Event Management (SIEM) dalam Meningkatkan Keamanan Jaringan,” *Jurnal Informatika*, vol. 8, no. 1, pp. 45–56, 2024.

[3] A. Patel, “SQL Injection Vulnerabilities and Prevention,” *Journal of Web Security*, vol. 3, no. 4, pp. 25–34, 2021.

[4] M. Sofiyan Hadi et al., “Implementasi SIEM untuk Deteksi dan Analisa Insiden Keamanan pada Web Server,” *Jurnal Keamanan Siber*, vol. 6, no. 2, pp. 78–89, 2023.

[5] SE Prasetyo et al., “Sistem Keamanan Website dari Serangan DoS, SQL Injection, XSS Menggunakan Web Application Firewall,” *Prosiding Nasional*, pp. 112–120, 2024.

[6] D.R. Sijabat et al., “Perancangan SIEM untuk Mendeteksi Insiden pada Situs Web,” *Jurnal Sistem Informasi*, vol. 4, no. 1, pp. 33–42, 2023.

[7] B. Smith, “Security Information and Event Management: Concepts and Implementation,” *International Journal of Cybersecurity*, vol. 12, no. 3, pp. 55–66, 2020.

[8] Wazuh Team, “Wazuh Open Source Security Platform Documentation,” Wazuh Inc., 2024. [Online]. Available: https: //vazuh.com/docs