ROUTERS: Jurnal Sistem dan Teknologi Informasi
Vol. 1, No. 1, Juli 2022, pp. xx-xx
ISSN: xxxx-xxxx, DOI: 10.xxxxx/routers.v0i0.xxxx

ANALISIS EFEKTIVITAS ATURAN WAZUH DALAM MENDETEKSI ANCAMAN WEB APLIKASI YANG MENGHASILKAN ERROR CODE DENGAN MENGIKUTI PEDOMAN OWASP TOP 10

Fatih Rizky Darmawan [1], Hamdani Arif [2]
[1] Rekayasa Keamanan Siber, Politeknik Negeri Batam
[2] Rekayasa Keamanan Siber, Politeknik Negeri Batam

INFORMASI ARTIKEL

Diterima 28 Juli 2021
Direvisi 28 Agustus 2021
Diterbitkan 28 September 2021

Kata kunci:
Wazuh, SIEM, OWASP Top 10, Error Code, Alerte Fatigue, Kustominasi Aturan, DVWA

ABSTRAK

Keamanan aplikasi web menjadi prioritas utama dalam ekosistem digital seiring eskalasi aktivitas pemindaian agresif dan eksploitasi celah keamanan yang tercantum dalam pedoman OWASP Top 10. Aktivitas serangan web secara konsisten meninggalkan jejak telemetri berupa status kode kesalahan HTTP (error code) pada log server. Meskipun platform Security Information and Event Management (SIEM) open-source seperti Wazuh memiliki kemampuan normalisasi log, aturan bawaan (default rules) memicu fenomena alert fatigue karena merekam setiap kode kesalahan secara individu sebagai single event alert. Penelitian ini bertujuan untuk menganalisis dan mengoptimalkan efektivitas deteksi Wazuh melalui pengembangan aturan kustom (custom tuning rules) berbasis korelasi kriteria frequency threshold. Pengujian dan simulasi serangan komprehensif dilakukan terhadap platform Damn Vulnerable Web Application (DVWA) dengan memuntahkan variasi insiden sukses dan gagal untuk taktik SQL Injection (SQLi), Local File Inclusion (LFI), dan Remote Code Execution (RCE). Metode tuning diimplementasikan melalui berkas konfigurasi XML fatih.xml dengan memisahkan serangan sukses pada level kritis instan (level="12") berstatus HTTP 200, serta meredam log noise status HTTP 302, 400, 404, dan 500 ke tingkat memori internal (level="1") sebelum diagregasikan ke dalam batas ambang frekuensi waktu. Hasil eksperimen menunjukkan bahwa aturan kustom Wazuh secara superior menekan alert fatigue dengan mengompresi volume log noise hingga lebih dari 90% menjadi hit alert bernilai tinggi, sekaligus meningkatkan visibilitas ancaman RCE yang sebelumnya tidak terdeteksi oleh aturan bawaan.

Analysis of the Effectiveness of Wazuh Rules in Detecting Web Application Threats Generating Error Codes Based on OWASP Top 10 Guidelines

ARTICLE INFO

Received July 28, 2021
Revised August 28, 2021
Published September 28, 2021

Keyword:
Wazuh, SIEM, OWASP Top 10, Error Codeh, Alerte Fatigue, Custom Rules , DVWA

ABSTRACT

Web application security has become a paramount concern in the digital ecosystem due to the escalation of aggressive scanning and exploitation activities targeting vulnerabilities listed in the OWASP Top 10 guidelines. Web attack behaviors consistently leave telemetry footprints in the form of HTTP error status codes within server logs. Although open-source Security Information and Event Management (SIEM) platforms like Wazuh offer robust log normalization capabilities, their default rules trigger alert fatigue by individualizing every generated error code into a single event alert. This research aims to analyze and optimize the detection effectiveness of Wazuh through the development of customized rules built upon frequency-based threshold correlation criteria. Comprehensive testing and attack simulations were executed against the Damn Vulnerable Web Application (DVWA) platform by bombarding it with variations of successful and failed incidents encompassing SQL Injection (SQLi), Local File Inclusion (LFI), and Remote Code Execution (RCE) tactics. The tuning methodology was implemented using a custom XML configuration file (fatih.xml) designed to segregate successful HTTP 200 attacks into instant critical thresholds (level="12") while muting redundant HTTP 302, 400, 404, and 500 noise logs into internal memory (level="1") prior to aggregate time-frequency calculations. Experimental results demonstrate that the customized Wazuh rules superiorly mitigate alert fatigue by compressing noise log volume by over 90% into high-value alerts, whilst simultaneously unlocking visibility into RCE threats that previously evaded default signature lookups.

1. PENDAHULUAN

Perkembangan teknologi informasi telah mendorong transformasi digital di berbagai sektor, termasuk pemerintahan, perbankan, pendidikan, kesehatan, hingga industri manufaktur. Sebagian besar layanan digital saat ini disediakan melalui aplikasi web karena kemudahan akses, fleksibilitas, serta kemampuan integrasi dengan berbagai sistem. Namun, meningkatnya ketergantungan terhadap aplikasi web juga diikuti oleh peningkatan risiko serangan siber yang semakin kompleks. Penyerang tidak lagi hanya menargetkan infrastruktur jaringan, tetapi juga mengeksploitasi kelemahan yang terdapat pada aplikasi web untuk memperoleh akses tidak sah, mencuri informasi sensitif, maupun mengganggu ketersediaan layanan. Kondisi tersebut menjadikan keamanan aplikasi web sebagai salah satu aspek yang sangat penting dalam menjaga keberlangsungan operasional organisasi serta melindungi aset informasi yang dimiliki [1][2].

Berdasarkan Verizon Data Breach Investigations Report (DBIR) 2025, eksploitasi aplikasi web masih menjadi salah satu penyebab utama terjadinya insiden keamanan informasi pada berbagai sektor industri. Laporan tersebut menunjukkan bahwa sebagian besar serangan diawali dengan aktivitas reconnaissance atau pemindaian terhadap target untuk mengidentifikasi layanan, direktori, maupun kerentanan yang dapat dieksploitasi. Aktivitas pemindaian tersebut umumnya dilakukan secara otomatis menggunakan bot maupun framework penetration testing sehingga mampu menghasilkan ribuan permintaan HTTP dalam waktu yang sangat singkat [1]. Fenomena serupa juga dilaporkan oleh Palo Alto Networks Unit 42, yang menyatakan bahwa hampir setengah dari aktivitas berbahaya pada lingkungan cloud berasal dari automated scanning dan exploitation terhadap layanan yang dapat diakses melalui internet [3].

Di Indonesia, ancaman terhadap aplikasi web juga menunjukkan tren peningkatan. Berdasarkan laporan tahunan Badan Siber dan Sandi Negara (BSSN), jutaan hingga ratusan juta aktivitas anomali masih terdeteksi setiap tahun, dengan kategori serangan yang didominasi oleh scanning, brute force, malware, serta eksploitasi layanan berbasis web. Target serangan tidak hanya berasal dari sektor pemerintahan, tetapi juga sektor keuangan, pendidikan, telekomunikasi, dan industri strategis lainnya. Tingginya aktivitas tersebut menunjukkan bahwa organisasi memerlukan mekanisme pemantauan keamanan yang mampu mendeteksi indikasi serangan secara cepat dan akurat sebelum berkembang menjadi insiden keamanan yang lebih besar [4].

Setiap aktivitas yang terjadi pada aplikasi web akan menghasilkan jejak digital dalam bentuk log yang disimpan oleh web server. Log tersebut berisi informasi mengenai alamat IP sumber, metode HTTP, Uniform Resource Identifier (URI), waktu akses, user-agent, serta status respons HTTP yang diberikan oleh server. Informasi tersebut memiliki nilai yang sangat penting karena dapat digunakan sebagai sumber bukti digital dalam proses monitoring keamanan, investigasi insiden, digital forensik, maupun threat hunting. Menurut NIST Special Publication 800-92, pengelolaan log yang baik merupakan salah satu komponen utama dalam membangun kemampuan deteksi dan respons terhadap insiden keamanan informasi [5].

Salah satu informasi penting yang terdapat pada web server log adalah HTTP Status Code. Berdasarkan standar RFC 9110 HTTP Semantics, status code digunakan untuk menunjukkan hasil pemrosesan suatu permintaan HTTP oleh server. Kode status seperti HTTP 200 menunjukkan permintaan berhasil diproses, sedangkan HTTP 302 menunjukkan pengalihan (redirect), HTTP 400 menunjukkan kesalahan permintaan dari sisi klien, HTTP 404 menunjukkan sumber daya tidak ditemukan, dan HTTP 500 menunjukkan terjadinya kesalahan pada sisi server [6]. Dalam konteks keamanan siber, kemunculan status code tersebut tidak hanya merepresentasikan kondisi operasional aplikasi, tetapi juga dapat menjadi indikator awal adanya aktivitas scanning, probing, brute force, fuzzing, maupun eksploitasi terhadap aplikasi web [7].

Untuk mengidentifikasi jenis ancaman yang menyerang aplikasi web, komunitas keamanan informasi secara luas menggunakan OWASP Top 10 sebagai salah satu acuan utama. OWASP Top 10 merupakan daftar sepuluh kategori kerentanan aplikasi web yang paling kritis berdasarkan data insiden keamanan dari berbagai organisasi di seluruh dunia. Beberapa kategori yang paling sering dieksploitasi antara lain Broken Access Control, Cryptographic Failures, Injection, Security Misconfiguration, Vulnerable and Outdated Components, serta Identification and Authentication Failures [8]. Kerangka kerja tersebut banyak digunakan sebagai standar dalam proses pengujian keamanan aplikasi maupun sebagai acuan dalam pengembangan mekanisme deteksi pada Security Operations Center (SOC).

Seiring meningkatnya volume log yang dihasilkan oleh aplikasi web, organisasi membutuhkan solusi yang mampu mengumpulkan, menormalisasi, mengkorelasikan, dan menganalisis log secara otomatis. Salah satu teknologi yang banyak digunakan adalah Security Information and Event Management (SIEM). SIEM memungkinkan organisasi mengintegrasikan berbagai sumber log ke dalam satu platform sehingga mempermudah proses monitoring, korelasi kejadian, serta investigasi insiden keamanan. Selain itu, SIEM juga mampu menghasilkan alert secara real-time ketika ditemukan aktivitas yang sesuai dengan aturan deteksi yang telah ditentukan [9], [10].

Salah satu SIEM berbasis open source yang banyak digunakan adalah Wazuh. Platform ini menyediakan berbagai kemampuan keamanan, seperti log collection, file integrity monitoring (FIM), vulnerability detection, security configuration assessment, malware detection, dan security event correlation. Wazuh menggunakan mekanisme rules engine berbasis XML untuk menganalisis setiap log yang diterima dari agen maupun sumber log lainnya. Fleksibilitas tersebut memungkinkan administrator membuat aturan deteksi yang disesuaikan dengan kebutuhan organisasi sehingga mampu meningkatkan efektivitas proses monitoring keamanan [11], [12].

Meskipun demikian, penggunaan default rules pada Wazuh masih memiliki beberapa keterbatasan. Aturan bawaan cenderung memperlakukan setiap log sebagai peristiwa yang berdiri sendiri (single event). Ketika terjadi aktivitas scanning secara masif, ribuan log HTTP 302, HTTP 400, HTTP 404, maupun HTTP 500 dapat menghasilkan ribuan alert yang sebenarnya memiliki konteks serangan yang sama. Kondisi tersebut menyebabkan munculnya fenomena alert fatigue, yaitu keadaan ketika analis SOC menerima terlalu banyak alert sehingga sulit membedakan alert yang benar-benar penting dengan alert bernilai rendah. Akibatnya, kemungkinan terjadinya false positive meningkat, sementara ancaman yang lebih kritis berpotensi terlewatkan karena tertutup oleh banyaknya alert yang muncul [13].

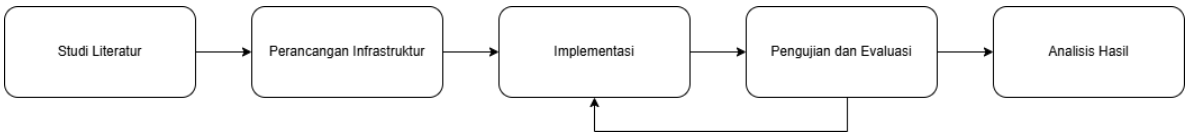
Berbagai penelitian sebelumnya telah menunjukkan bahwa penerapan teknik korelasi log, threshold frequency, serta rule tuning mampu meningkatkan kualitas deteksi pada platform SIEM. Dengan mengelompokkan sejumlah event yang memiliki karakteristik serupa ke dalam satu alert, jumlah alert dapat dikurangi secara signifikan tanpa menghilangkan informasi penting mengenai aktivitas serangan. Selain meningkatkan efisiensi monitoring, pendekatan tersebut juga membantu analis SOC memprioritaskan penanganan terhadap ancaman yang memiliki tingkat risiko lebih tinggi [14], [16].

Berdasarkan permasalahan tersebut, penelitian ini mengusulkan pengembangan custom rules pada Wazuh menggunakan mekanisme frequency threshold correlation. Aturan yang dikembangkan bertujuan untuk membedakan antara aktivitas serangan yang benar-benar berhasil dengan aktivitas scanning atau probing yang hanya menghasilkan noise. Log HTTP 302, HTTP 400, HTTP 404, dan HTTP 500 akan diperlakukan sebagai indikator aktivitas berulang yang baru menghasilkan alert ketika jumlah kemunculannya telah melewati ambang batas tertentu. Sebaliknya, aktivitas eksploitasi yang menghasilkan HTTP 200 sebagai indikasi keberhasilan serangan akan langsung menghasilkan alert dengan tingkat keparahan tinggi sehingga dapat segera ditindaklanjuti oleh analis SOC.

Untuk mengevaluasi efektivitas pendekatan tersebut, penelitian ini menggunakan Damn Vulnerable Web Application (DVWA) sebagai lingkungan pengujian. Beberapa skenario serangan yang mengacu pada kategori OWASP Top 10, seperti SQL Injection (SQLi), Local File Inclusion (LFI), dan Remote Code Execution (RCE), disimulasikan menggunakan skrip otomatis sehingga mampu menghasilkan variasi log dalam jumlah besar. Selanjutnya, performa antara aturan bawaan Wazuh dan aturan hasil tuning dibandingkan berdasarkan jumlah alert yang dihasilkan, tingkat reduksi noise, kemampuan mendeteksi serangan yang berhasil, serta analisis confusion matrix untuk mengukur peningkatan nilai True Positive, False Positive, False Negative, dan True Negative. Hasil penelitian ini diharapkan dapat memberikan kontribusi terhadap pengembangan mekanisme deteksi pada platform Wazuh sehingga mampu meningkatkan efektivitas monitoring keamanan aplikasi web sekaligus mengurangi fenomena alert fatigue pada lingkungan Security Operations Center (SOC).

2. METODE

Penelitian ini menggunakan metode eksperimen dengan tahapan terstruktur yang meliputi studi literatur, perancangan infrastruktur, implementasi sistem, pengujian dan evaluasi, serta analisis hasil. Adapun tahapan – tahapan tersebut pada Gambar 1 di bawah ini:



Gambar 1 Langkah Penelitian

2.1. Studi Literatur

Pada tahapan ini, dilakukan proses pengumpulan, penelaahan, dan kompilasi landasan teori dari berbagai sumber ilmiah dan dokumentasi teknis otoritatif yang relevan dengan objek riset. Referensi utama yang digunakan mencakup standar ancaman aplikasi web OWASP Top 1, kerangka kerja MITRE ATT&CK Framework[15], regulasi NIST SP 800-92 untuk Log Management, serta dokumentasi resmi sintaks ruleset Wazuh Manager. Dokumen-dokumen tersebut digunakan sebagai fondasi akademik dalam menganalisis akar masalah alert fatigue dan menyusun arsitektur aturan kustom.

2.2. Perancangan Infrastruktur

Tahapan ini berfokus pada perancangan arsitektur laboratorium keamanan siber terisolasi untuk mendukung proses pengumpulan log. Infrastruktur dirancang menggunakan platform hypervisor Proxmox VE yang memuat dua buah mesin virtual (VM) utama. VM pertama bertindak sebagai target serangan (Server Korban) yang menjalankan web server Apache dan aplikasi rentan Damn Vulnerable Web Application(DVWA). VM kedua dikonfigurasi sebagai Server Monitoring (SIEM Wazuh Manager) yang bertindak sebagai pusat pengumpul, pengurai, dan pengorelasian data telemetri log dari agen. Implementasi Infrastruktur yang telah dirancang akan diterapkan dengan menjalankan teknologi Wazuh dan DVWA.

2.3. Pengujian dan Evaluasi

Pada tahapan ini, simulasi serangan siber dan pemicuan kebisingan trafik (noise generation) diluncurkan secara otomatis dari mesin penyerang (Kali Linux). Pengujian dieksekusi menggunakan skrip Bash otomatisasi `stress_test_wazuh.sh` dengan parameter inter-jeda waktu yang sangat rapat (`DELAY=0.02`) dan batas perulangan pasti (`JUMLAH_TEMBAKAN=25`). Kategori skenario pengujian dibagi menjadi tiga fase logis:

1. Fase Eksploitasi Sukses (Dengan Autentikasi - HTTP 200/500): Menembakkan serangan SQL Injection (SQLi), Local File Inclusion (LFI), dan Remote Code Execution (RCE) untuk menguji keandalan Instant Alerting.
2. Fase Serangan Gagal (Tanpa Autentikasi - HTTP 302): Mengirimkan serangan tanpa menyertakan session cookie login untuk memancing respons pengalihan default aplikasi.
3. Fase Variasi Kebisingan Sistem (HTTP 400, 404, 503): Menembak file simulasi acak dan memanipulasi header sintaks guna memicu respons eror murni web server.

2.4. Analisis Hasil

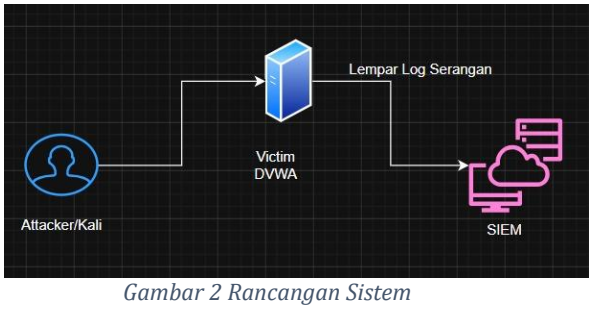
Data volume alert yang terekam pada dasbor SIEM Wazuh dikumpulkan dan diklasifikasikan ke dalam tabel matriks data. Hasil performa aturan kustom dievaluasi secara komparatif terhadap aturan bawaan (default rules) dengan mengukur tingkat reduksi log sampah (Noise Reduction Rate) serta peningkatan sensitivitas visualisasi ancaman kritikal di dashboard analis SOC.

3. HASIL DAN PEMBAHASAN

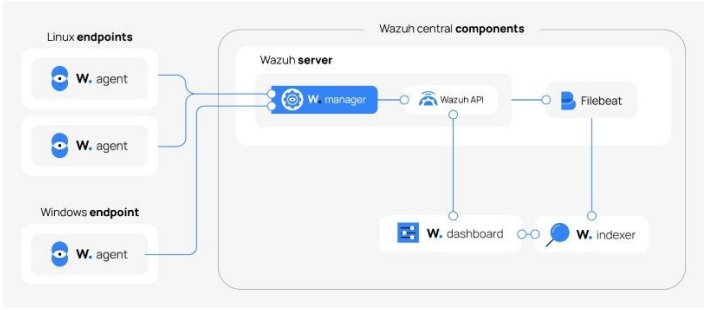
3.1. Rancangan Sistem

Palo Alto Networks Unit 42 Cloud Threat Report Pada penelitian ini, eksperimen dilakukan menggunakan teknologi mesin virtual yang di pasangkan Proxmox sebagai *hypervisor,* seperti terlihat pada Gambar 2 yang merupakan rancangan sistem yang digunakan pada penelitian ini. Selanjutnya, sistem melibatkan penggunaan pendekatan *Infrastructure as Code* (IaC) dengan memanfaatkan Terraform untuk mengotomatisasi penyediaan server. Setelah server berhasil di sediakan, proses *security hardening* diterapkan menggunakan Ansible. Alur dari proses tersebut, dapat dilihat pada

Gambar 3.



Gambar 2 Rancangan Sistem



Gambar 3 Rancangan SIEM

Adapun teknologi yang digunakan penulis pada penelitian ini tertera pada Tabel 1 dibawah ini.

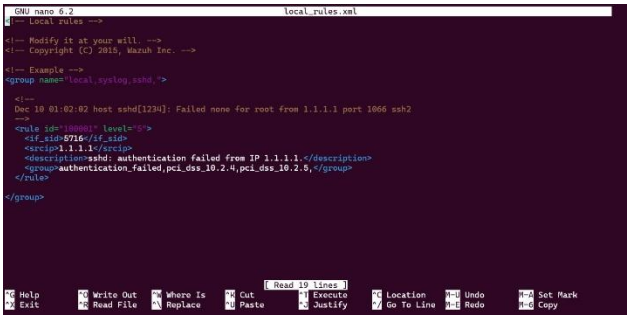
Tabel 1 Teknologi yang digunakan

Teknologi	Versi
wazuh	4.14.16
filebeat	9.4.3
apache	2.4.68
kali linux	2025.2
Ubuntu Server	24.4.02

3.2. Hasil Implementasi

3.2.1 Implementasi Default Rules Wazuh

Implementasi awal dilakukan menggunakan rules bawaan Wazuh tanpa melakukan modifikasi terhadap ruleset. Seluruh log Apache diproses menggunakan decoder bawaan Wazuh kemudian dicocokkan terhadap rules yang terdapat pada direktori /var/ossec/ruleset/lrules. Hasil implementasi ini digunakan sebagai kondisi baseline untuk mengetahui karakteristik alert yang dihasilkan sebelum dilakukan proses tuning. Proses pembuatan/implement Agent Pada Wazuh Sebagai Korban sebagai *template* dapat dilihat pada Gambar 4 serta Gambar 5.

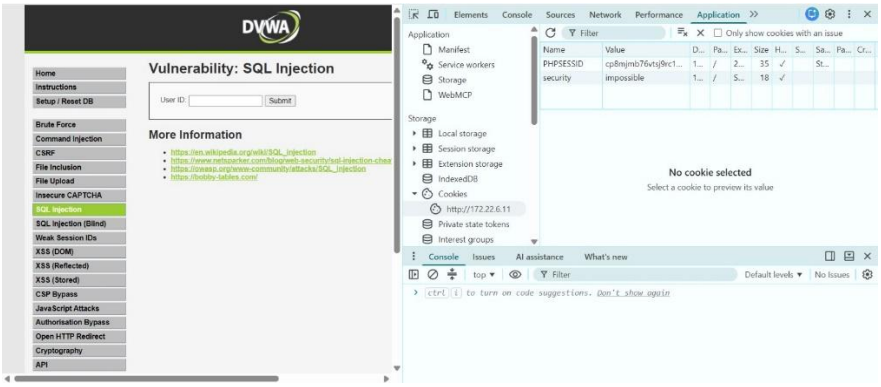


Gambar 4 Rules Wazuh Default

3.2.2 Implementasi DVWA Sebagai Target

DVWA diimplementasikan pada sebuah mesin virtual yang menjalankan sistem operasi Ubuntu Server 24.04 LTS dengan layanan Apache2 sebagai web server, PHP sebagai bahasa pemrograman aplikasi, dan

MariaDB/MySQL sebagai basis data. Seluruh layanan dikonfigurasi agar dapat menghasilkan access log dan error log Apache yang kemudian dikirimkan menuju Wazuh Manager menggunakan Wazuh Agent. Arsitektur ini memungkinkan setiap aktivitas yang dilakukan oleh penyerang dapat direkam dan dianalisis oleh mekanisme rule engine Wazuh.



Gambar 5 Implement DVWA Sebagai Victim

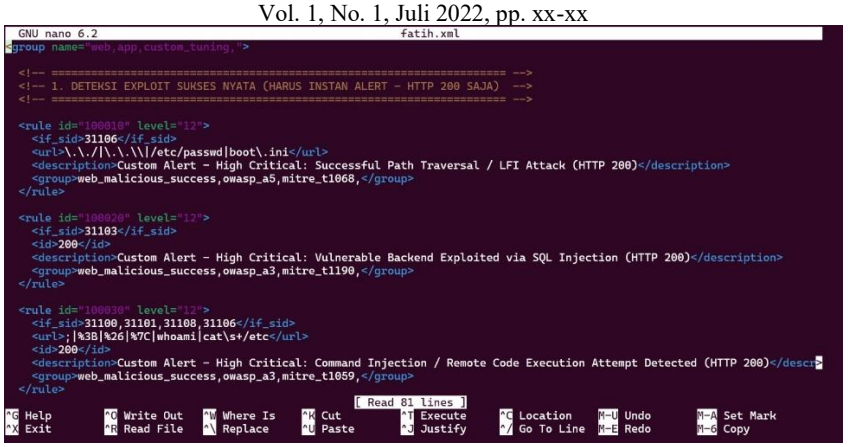
Setelah implementasi DVWA selesai dilakukan, tahap berikutnya adalah mengembangkan dan menerapkan custom tuning rules pada Wazuh. Aturan ini dirancang untuk mengurangi alert fatigue melalui mekanisme frequency threshold correlation tanpa mengurangi kemampuan sistem dalam mendeteksi aktivitas serangan terhadap aplikasi web. Pembahasan mengenai implementasi aturan tersebut dijelaskan pada Subbab 3.2.3 Implementasi Custom Rules.

3.2.3. Implementasi Custom Rules Wazuh

Setelah implementasi default rules selesai dilakukan, tahap berikutnya adalah mengembangkan custom tuning rules pada Wazuh menggunakan berkas konfigurasi fatih.xml. Pengembangan aturan ini memanfaatkan mekanisme custom rule engine yang disediakan oleh Wazuh, seperti penggunaan parent rule, event correlation, serta parameter frequency dan timeframe untuk mengelompokkan beberapa event yang memiliki karakteristik serupa menjadi satu alert. Pendekatan tersebut mengacu pada dokumentasi resmi Wazuh mengenai mekanisme rule correlation dan custom rules. [1] [2]

Aturan ini dibuat untuk mengurangi jumlah alert yang tidak relevan (alert fatigue) dengan menerapkan mekanisme frequency threshold correlation, sehingga beberapa log dengan karakteristik yang sama dapat digabungkan menjadi satu security alert. Konsep korelasi alert tersebut banyak digunakan pada sistem SIEM untuk mengurangi volume alert berulang dan meningkatkan efektivitas analisis keamanan. [2], [5]
Selain itu, aturan juga diklasifikasikan berdasarkan tingkat risiko dengan memberikan level 12 untuk indikasi eksploitasi yang diprioritaskan dan level 1 sebagai collector rule untuk event HTTP yang akan dikorelasikan menggunakan frequency threshold. Penentuan kategori serangan pada aturan ini mengacu pada pedoman OWASP Top 10:2021, sedangkan penggunaan HTTP status code mengacu pada standar HTTP Semantics (RFC 9110). [3], [4]

Gambar 6 menunjukkan implementasi *custom rules* yang digunakan pada penelitian ini. Aturan tersebut menjadi pengembangan terhadap *default rules* Wazuh untuk mengevaluasi efektivitas deteksi ancaman web aplikasi berdasarkan HTTP status code. Perancangan *custom rules* mengacu pada mekanisme *rule engine* Wazuh dengan penyesuaian terhadap skenario serangan yang mengikuti kategori OWASP Top 10 sehingga mampu membedakan antara aktivitas eksploitasi dan *noise alert*. [1], [3]



Gambar 6 Hasil Pembuatan Tuning Rules

Tabel 2 Implementasi Custom Rules Wazuh

Rule ID	HTTP Status	Level	Frequency	Timeframe	Fungsi
100020	HTTP 200	12	-	-	Deteksi SQL Injection berhasil
100010	HTTP 200	12	-	-	Deteksi Local File Inclusion berhasil
100030	HTTP 200	12	-	-	Deteksi Remote Code Execution berhasil
100051	HTTP 302	6	10	60 detik	Korelasi redirect berulang
100071	HTTP 400	6	10	60 detik	Mengurangi alert request tidak valid
100081	HTTP 404	6	10	60 detik	Korelasi directory scanning
100061	HTTP 500	6	5	60 detik	Korelasi error internal server
100091	HTTP 503	6	5	60 detik	Korelasi service unavailable

3.2.3 Kode Script

Script pengujian dikembangkan menggunakan perintah curl dengan beberapa variasi skenario, seperti SQL Injection (SQLi), Local File Inclusion (LFI), Remote Code Execution (RCE), serta permintaan yang menghasilkan HTTP 302, 400, 404, 500, dan 503. Setiap skenario dijalankan secara otomatis dengan jumlah pengulangan (loop) dan jeda waktu (delay) tertentu untuk mensimulasikan aktivitas web scanning maupun percobaan eksploitasi yang umum terjadi pada lingkungan nyata. Seluruh log yang dihasilkan kemudian dikirim menuju Wazuh Manager melalui Wazuh Agent untuk dianalisis menggunakan default rules dan custom tuning rules.

Gambar 7 menunjukkan potongan script yang digunakan dalam proses pengujian. Hasil eksekusi script tersebut menjadi dasar dalam mengevaluasi perbedaan jumlah alert yang dihasilkan oleh default rules dan custom tuning rules pada tahap pengujian selanjutnya

Gambar 7 Potongan Script Untuk Attacker Menguji DVWA

Gambar 8 menunjukkan hasil eksekusi script pengujian menggunakan default rules. Pada kondisi ini setiap HTTP request yang memenuhi kriteria aturan akan menghasilkan alert secara individual, sehingga jumlah alert yang diterima sebanding dengan jumlah permintaan yang dikirimkan. Hasil tersebut kemudian dibandingkan dengan custom tuning rules untuk mengetahui efektivitas mekanisme frequency threshold correlation dalam mengurangi *alert fatigue*

Top values of data.id	Count of records
200	25
302	25
400	25
404	25
500	25
503	25

Gambar 8 Hasil Eksekusi Script

3.3. Hasil Pengujian Default Rules

Pengujian pertama dilakukan menggunakan rules bawaan Wazuh. Setiap skenario serangan dijalankan sebanyak 25 kali dengan interval 0,02 detik. Seluruh request menghasilkan alert secara individual sehingga jumlah alert sama dengan jumlah request yang dikirimkan.

Tabel 3 Hasil Uji Rules Default

Error Code Rules Default	Result Setelah di uji
200	25
302	25
400	25
404	25
500	25
503	25

Berdasarkan hasil pengujian yang ditunjukkan pada Tabel 3, setiap HTTP request yang memenuhi kriteria default rules menghasilkan satu security alert. Seluruh kategori HTTP status code, yaitu HTTP 200, 302, 400, 404, 500, dan 503, masing-masing menghasilkan 25 alert, sesuai dengan jumlah permintaan yang dikirimkan. Kondisi ini menunjukkan bahwa default rules Wazuh masih memproses setiap log sebagai single event alert tanpa melakukan korelasi terhadap aktivitas yang memiliki pola serupa.

Akibatnya, ketika terjadi aktivitas web scanning atau percobaan eksploitasi secara berulang, jumlah alert yang dihasilkan meningkat secara signifikan. Kondisi tersebut berpotensi menimbulkan alert fatigue, yaitu situasi ketika analis Security Operations Center (SOC) menerima terlalu banyak alert dengan konteks serangan yang sama sehingga menyulitkan proses identifikasi ancaman yang benar-benar penting. Oleh karena itu, hasil pengujian default rules pada penelitian ini dijadikan sebagai acuan untuk mengevaluasi

efektivitas custom tuning rules dalam mengurangi jumlah alert tanpa mengurangi kemampuan sistem dalam mendeteksi aktivitas serangan.

3.4. Hasil Pengujian Custom Rules

Setelah custom tuning rules diterapkan, pengujian kembali dilakukan menggunakan skenario serangan yang sama seperti pada default rules. Pengujian ini bertujuan untuk mengetahui efektivitas aturan yang telah dimodifikasi dalam mengurangi jumlah alert yang tidak relevan tanpa mengurangi kemampuan deteksi terhadap aktivitas serangan.

Berdasarkan Tabel 3, jumlah alert pada HTTP 302, 400, 404, 500, dan 503 mengalami penurunan dibandingkan default rules. Hal ini menunjukkan bahwa mekanisme frequency threshold correlation berhasil menggabungkan beberapa event dengan karakteristik yang sama menjadi satu alert. Sementara itu, serangan yang menghasilkan HTTP 200 tetap diprioritaskan sebagai critical alert sehingga aktivitas yang berhasil dieksploitasi masih dapat terdeteksi dengan baik.

Hasil tersebut menunjukkan bahwa custom tuning rules mampu mengurangi alert fatigue serta meningkatkan efektivitas monitoring pada Wazuh dengan menghasilkan alert yang lebih ringkas dan relevan bagi analis Security Operations Center (SOC)

Error Code Rules Default	Result Setelah di uji
200	50
302	3
400	2
404	2
500	2
503	2

Table.5 Hasil Uji untuk Tuning Rules

Gambar.9 Dashboard

3.5. Analisis Perbandingan

Berdasarkan hasil pengujian, default rules menghasilkan alert untuk setiap HTTP request yang memenuhi kondisi aturan sehingga jumlah alert meningkat sebanding dengan jumlah permintaan yang dikirimkan. Setelah dilakukan implementasi custom tuning rules, jumlah alert mengalami penurunan pada HTTP 302, HTTP 400, HTTP 404, HTTP 500, dan HTTP 503 karena beberapa event berhasil dikorelasikan menggunakan mekanisme frequency threshold. Sebaliknya, serangan yang menghasilkan HTTP 200 tetap diprioritaskan sebagai critical alert karena menunjukkan aktivitas eksploitasi yang berhasil dilakukan.

Tabel 4 Tabel Hasil Perbandingan

Error Code	Default	Custom	Pengurangan
200	25	50	-
302	25	3	91,7%
400	25	2	92%
404	25	2	92%
500	25	2	92%
503	25	2	92%

3.6. Analisis Dampak Custom Tuning Rules terhadap Alert Fatigue

Penerapan custom tuning rules memberikan dampak yang signifikan terhadap pengurangan alert fatigue pada Wazuh Dashboard. Pada kondisi default rules, setiap HTTP request yang memenuhi aturan akan menghasilkan satu alert, sehingga aktivitas web scanning maupun percobaan eksploitasi menghasilkan alert dalam jumlah besar dengan konteks yang sama. Kondisi tersebut menyebabkan analis Security Operations Center (SOC) harus melakukan proses analisis terhadap banyak alert yang sebenarnya berasal dari satu aktivitas serangan.

Setelah diterapkan mekanisme frequency threshold correlation, beberapa event yang memiliki karakteristik serupa berhasil digabungkan menjadi satu alert. Dengan demikian, jumlah alert yang ditampilkan pada dashboard menjadi lebih sedikit tanpa mengurangi kemampuan sistem dalam mendeteksi aktivitas serangan. Hasil ini menunjukkan bahwa proses rule tuning mampu meningkatkan efisiensi proses monitoring serta membantu analis SOC memprioritaskan alert yang memiliki tingkat risiko lebih tinggi.

Tabel 5 Total Keseluruhan

Parameter	Default	Custom
Total Alert	150	61
Noise Alert	125	11
Critical Alert	25	50

4. KESIMPULAN

Berdasarkan hasil implementasi, pengujian, dan analisis yang telah dilakukan, dapat disimpulkan bahwa penerapan custom tuning rules berbasis frequency threshold correlation pada Wazuh mampu meningkatkan efektivitas deteksi ancaman aplikasi web dibandingkan dengan default rules. Pendekatan yang diusulkan berhasil mengurangi jumlah noise alert, meminimalkan fenomena alert fatigue, serta meningkatkan visibilitas terhadap aktivitas serangan yang sesuai dengan kategori OWASP Top 10. Dengan demikian, penelitian ini diharapkan dapat menjadi referensi dalam pengembangan mekanisme deteksi berbasis SIEM serta mendukung peningkatan efisiensi proses monitoring dan analisis keamanan pada lingkungan Security Operations Center (SOC).

DAFTAR PUSTAKA

[1] Verizon, “2025 Data Breach Investigations Report,” 2025. [Online]. Available: https://www.verizon.com/business/resources/reports/dbir/
[2] I. Institute fro Business Value, “IBM X-Force 2025 Threat Intelligence Index.”
[3] Palo Alto Networks, “Unit 42 Cloud Threat Report,” 2022.
[4] Badan Siber dan Sandi Negara, “Laporan Tahunan Keamanan Siber Indonesia 2023,” Jakarta, 2023.
[5] National Institute of Standards and Technology, “Special Publication 800-92: Guide to Computer Security Log Management,” 2006.
[6] IETF, “RFC 9110: HTTP Semantics,” 2022. [Online]. Available: https://www.rfc-editor.org/rfc/rfc9110
[7] Internet Engineering Task Force (IETF), “RFC 9110: HTTP Semantics,” 2022.
[8] O. Foundation, “OWASP Top 10: The Ten Most Critical Web Application Security Risks,” 2021. [Online]. Available: https://owasp.org/Top10/
[9] National Institute of Standards and Technology, “Special Publication 800-61 Revision 2: Computer Security Incident Handling Guide,” 2012.
[10] O. Abouabdalla, H. El-Taj, A. Manasrah, and S. Ramadass, “False positive reduction in intrusion detection system: A survey,” in *2009 2nd IEEE International Conference on Broadband Network & Multimedia Technology*, 2009, pp. 463–466. doi: 10.1109/ICBNMT.2009.5348536.

[11] Wazuh Inc., "Wazuh Documentation," 2025. [Online]. Available: https://documentation.wazuh.com/current/
[12] Wazuh Inc., "Wazuh User Manual - Ruleset Reference," 2026. [Online]. Available: https://documentation.wazuh.com/current/
[13] T. Ahmed, A. Shah, M. Kolla, and R. Yellasiri, "Reduction of Alert Fatigue using Extended Isolation Forest," in *2021 International Conference on Forensics, Analytics, Big Data, Security (FABS)*, 2021, pp. 1–5. doi: 10.1109/FABS52071.2021.9702617.
[14] B.-D. Kwon and A. J. Rudman, "Correlation of geologic logs with spectral methods," *Journal of the International Association for Mathematical Geology*, vol. 11, no. 4, pp. 373–390, 1979, doi: 10.1007/BF01029295.
[15] MITRE, "MITRE ATT&CK Framework," 2025. [Online]. Available: https://attack.mitre.org/