INTEGRASI *WAZUH SIEM* DENGAN *MODSECURITY* DAN *VIRUS TOTAL* MENGGUNAKAN *NIST FRAMEWORK* UNTUK MENDETEKSI SERANGAN *WEBSITE*

Rizki Nurul Fahmi, Rudi Hartono, Dede Syahrul Anwar
Teknik Informatika, Universitas Perjuangan Tasikmalaya
Jalan Pembela Tanah Air (PETA) No. 177 Kota Tasikmalaya, Indonesia
rizkinurulfahmi@gmail.com

ABSTRAK

Peningkatan penggunaan *website* di Indonesia berpotensi menimbulkan ancaman siber yang mencakup aktivitas *ransomware*, *defacement website*, dan *malware*. Kurangnya sentralisasi dalam sistem keamanan siber dapat memperlambat deteksi ancaman, menyebabkan respons tidak terkoordinasi, serta meningkatkan risiko celah keamanan. Penelitian ini bertujuan untuk mengatasi permasalahan tersebut dengan mengintegrasikan *Wazuh*, *ModSecurity*, dan *Virus Total* untuk mendeteksi ancaman siber berdasarkan kerentanan *OWASP Top* 10 tahun 2021 yang terdapat pada *local website*. Penelitian ini dilakukan secara eksperimental menggunakan *NIST Framework* yang disertai dengan pengumpulan data dan evaluasi. Eksperimen dilaksanakan dalam *local environment*, di mana penulis mengintegrasikan *Wazuh* dengan *ModSecurity* dan *Virus Total*. Setelah itu, dilakukan *website penetration testing* terhadap *local website*. Hasil penelitian ini menunjukkan bahwa integrasi ketiga alat tersebut mampu mendeteksi 10 dari 17 jenis serangan terhadap *local website*, yaitu*: File Inclusion Attack, SQL Injection, Blind SQL Injection, Command Injection, Reflected XSS, Stored XSS, DOM-Based XSS, Open HTTP Redirect, File Upload, dan Brute Force*. Dengan demikian, sistem integrasi ini dapat memberikan perlindungan tambahan bagi *website* dari serangan siber.

Kata kunci: *SIEM, OWASP, NIST, Wazuh, ModSecurity, Virus Total*

1. PENDAHULUAN

Ketergantungan yang semakin tinggi pada *website* di Indonesia telah menjadikan mereka sasaran utama serangan siber. Sepanjang tahun 2023, Badan Siber dan Sandi Negara (BSSN) mencatat 403.990.813 anomali trafik, termasuk aktivitas *ransomware* dan *web defacement* [1]. Seiring dengan meningkatnya digitalisasi organisasi, risiko keamanan *website* pun bertambah [2], memicu lonjakan ancaman dari *hacker*, *malware*, dan *phishing* karena itu, praktik keamanan siber sangat penting untuk melawan ancaman tersebut [3].

Salah satu pendekatan yang menjanjikan dalam menghadapi ancaman ini adalah integrasi *Wazuh*, *ModSecurity*, dan *Virus Total*. *Wazuh* merupakan platform keamanan berbasis host yang menawarkan pemantauan dan respons insiden secara menyeluruh [4]. *ModSecurity* adalah *firewall* aplikasi *web open source* yang memeriksa lalu lintas *web* secara *real-time* untuk mencegah serangan seperti *XSS* dan *SQL Injection* [5], sedangkan *Virus Total* menyediakan intelijen ancaman dengan analisis terhadap *URL*, dan *file* [6].

Penelitian sebelumnya menunjukkan keterbatasan dalam pendekatan keamanan *website*. Ayunda et al. (2021) meneliti penerapan *ModSecurity*, namun kurang efektif dalam menangani ancaman *malware* dan serangan sistem [7]. Azzah Shafiyyah (2024) dan Faruq Aziz Saputra (2024) membahas penggunaan *Wazuh*muntuk deteksi ancaman siber, tetapi belum mengintegrasikan alat tambahan seperti *ModSecurity* dan *Virus Total* untuk cakupan perlindungan yang lebih luas [8].

Penelitian ini menggunakan *NIST Framework* sebagai metodologi, yang mencakup lima tahap utama: identifikasi, proteksi, deteksi, respons, dan pemulihan [9], serta didukung oleh studi literatur dan observasi. Tujuan utama penelitian ini adalah untuk mengintegrasikan *Wazuh*, *ModSecurity*, dan *Virus Total* dalam satu sistem keamanan, serta melakukan eksperimen dalam *virtual environment* terhadap kemampuannya dalam mendeteksi serangan berdasarkan kerentanan *OWASP Top* 10 tahun 2021, tidak termasuk kategori A09 dan A10. Serangan-serangan tersebut meliputi *Broken Access Control* (A01) yang dapat menyebabkan pengguna mendapatkan akses tidak sah ke data atau fungsi yang seharusnya dibatasi, *Cryptographic Failures* (A02) yang dapat mengakibatkan kebocoran data sensitif akibat penggunaan enkripsi yang lemah atau tidak tepat; *Injection* (A03) seperti *SQL injection*, yang dapat dimanfaatkan untuk mengambil alih basis data aplikasi, *Insecure Design* (A04) yang mencakup perancangan sistem tanpa mempertimbangkan aspek keamanan, membuka celah bagi berbagai eksploitasi, *Security Misconfiguration* (A05) yang timbul akibat pengaturan sistem yang tidak aman, seperti *default credentials* atau *port* yang terbuka, Vulnerable and Outdated Components (A06) yang dapat dieksploitasi karena komponen perangkat lunak tidak diperbarui atau memiliki celah keamanan yang diketahui; *Identification and Authentication Failures* (A07) yang dapat memungkinkan serangan *brute force* atau

pencurian kredensial, *Software and Data Integrity Failures* (A08) yang dapat dimanfaatkan untuk menyisipkan kode berbahaya melalui pembaruan atau *dependency* yang tidak tervalidasi. Selanjutnya dilakukan juga evaluasi terhadap efektivitas integrasi ketiga alat dalam meningkatkan keamanan *website* dan merumuskan rekomendasi pencegahan yang efektif terhadap serangan siber.

2. TINJAUAN PUSTAKA

2.1. *Security Information and Event Management* (*SIEM*)

SIEM adalah sistem yang mengumpulkan dan menganalisis data keamanan secara terpusat untuk mendeteksi ancaman dan insiden. Teknologi ini memungkinkan analisis peristiwa secara *real-time* dengan mengkorelasikan *log* dari berbagai sumber [10]. Selain itu, *SIEM* mencakup normalisasi data, *alert*, serta respons insiden otomatis (Rangga A., 2024) [11].

2.2. *Web Application Firewall (WAF)*

WAF merupakan lapisan perlindungan yang memantau, menyaring, dan memblokir lalu lintas data guna melindungi website dari serangan seperti *SQL injection* dan *XSS*. Teknologi ini bekerja dengan menerapkan aturan pada permintaan *HTTP* untuk mengidentifikasi aktivitas berbahaya [12]. Jika ditemukan ancaman, *WAF* akan memblokir akses dan memberi peringatan kepada tim keamanan.

2.3. *Website*

Website adalah sistem informasi berbasis *internet* yang memungkinkan akses informasi melalui alamat tertentu. Sebagai media komunikasi, *website* terdiri dari beberapa komponen, termasuk *server*, klien, jaringan, dan protokol seperti *HTTP* serta *TCP*/IP [13].

2.4. *Wazuh*

Wazuh adalah platform keamanan yang menyediakan fungsi *SIEM* dan *XDR* untuk mendeteksi ancaman dan memastikan kepatuhan keamanan. Arsitektur *Wazuh* terdiri dari *Indexer*, *Dashboard*, *Server*, dan *Agent* yang bekerja sama dalam mengumpulkan, menganalisis, serta mengelola peringatan keamanan [14].

2.5. *ModSecurity*

ModSecurity merupakan *Web Application Firewall* yang melindungi aplikasi web dari ancaman siber dengan menganalisis lalu lintas *HTTP*. Dengan fitur *Core Rule Set* (*CRS*), *ModSecurity* mendeteksi dan memblokir pola serangan umum, termasuk SQL *Injection* dan *XSS*. Sistem *logging* yang terintegrasi memungkinkan pelacakan aktivitas mencurigakan [15].

2.6. *Virus Total (VT)*

Virus Total adalah layanan analisis *malware* yang mengumpulkan data dari berbagai sumber untuk mendeteksi ancaman keamanan. Alat ini digunakan oleh peneliti dan profesional keamanan dalam mengevaluasi tingkat bahaya suatu *file*, *URL*, atau *IP address* [16].

2.7. *NIST Cybersecurity Framework (NIST CSF)*

NIST SP 1271 adalah panduan ringkas dari *NIST* yang membantu organisasi memulai atau meningkatkan program keamanan siber mereka dengan menerapkan *NIST Cybersecurity Framework*. Didalam dokumen ini terdapat lima fungsi utama *Identify*, *Protect*, *Detect*, *Respond*, dan *Recover* sebagai pendekatan sistematis untuk mengelola risiko keamanan siber secara menyeluruh [17]. *Framework* ini memungkinkan organisasi untuk menyesuaikan kebijakan keamanan berdasarkan perkembangan ancaman dan teknologi.

2.8. *OWASP Top 10*

OWASP Top 10 adalah daftar sepuluh risiko keamanan aplikasi web paling kritis yang disusun oleh komunitas *OWASP*. Versi 2021 mencakup kategori baru seperti Insecure *Design* dan *Software and Data Integrity Failures*, serta menyoroti ancaman utama seperti *Broken Access Control* dan *Injection*. Daftar ini bertujuan meningkatkan kesadaran dan membantu organisasi memprioritaskan upaya keamanan aplikasi mereka. [18].

2.9. Penelitian Terdahulu

Berdasarkan penelitian yang dilakukan oleh Yusuf Ardiansyah, M. Agus Sunandar, dan Yusuf Muhyidin (2023) dalam studi berjudul Implementasi Keamanan *Website* dengan Metode *Firewall Website* (*WAF*): Studi Kasus pada Web Desa Wantilan, penggunaan *WAF* terbukti efektif dalam memblokir serangan *SQL Injection* dan *DDoS*, serta mencegah perubahan konten yang tidak sah melalui konfigurasi *.htaccess* dan *Cloudflare* [19]. Namun, penelitian ini hanya mengandalkan *WAF* tanpa integrasi alat deteksi lainnya, sehingga kurang optimal dalam melakukan *monitoring* dan mendeteksi ancaman atau serangan yang lebih kompleks.

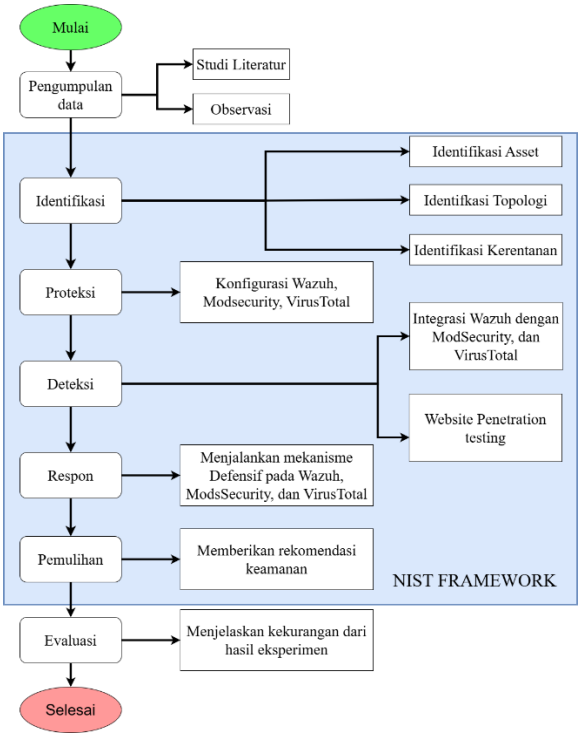
Sementara itu, penelitian oleh Rangga Aditya, Yusuf Muhyidin, dan Dayan Singasatia (2024) yang berjudul Implementasi *Security Information and Event Management* (*SIEM*) untuk *Monitoring* Keamanan *Server* Menggunakan *Wazuh* menunjukkan bahwa *Wazuh* efektif dalam mendeteksi serangan dengan tingkat kerentanan rendah seperti *DDOS* [20], tetapi kurang optimal dalam mendeteksi *SQL Injection*. Keterbatasan utama penelitian ini adalah tidak adanya integrasi dengan alat keamanan lain, sehingga sistem tidak dapat sepenuhnya menahan serangan.

Selain itu, penelitian Kantola Teemu (2022) dalam *Exploring Virus Total for Security Operations Alert Triage Automation* meneliti pemanfaatan *Virus Total* dalam mengotomatisasi *triase* peringatan keamanan melalui *SOAR*. Hasil penelitian menunjukkan peningkatan waktu respons insiden dan

akurasi dalam mengenali ancaman [21]. Namun, fokus penelitian ini hanya pada *Virus Total* untuk *IOC* (*Indicator of Compromise*) *enrichment*, tanpa integrasi tambahan seperti *WAF*, yang dapat meningkatkan deteksi ancaman secara lebih menyeluruh.

3. METODE PENELITIAN

Tahapan metodologi yang diterapkan pada penelitian ini menggunakan metode *NIST Framework* yang dipadukan dengan pengumpulan data melalui studi literatur dan observasi yang dapat dilihat pada Gambar 1 *Flowchart* metodologi penelitian.



Gambar 1. *Flowchart* metodologi penelitian

Metode penelitian yang digunakan adalah *NIST Cybersecurity Framework* yang terdiri dari lima tahap utama: Identifikasi, Proteksi, Deteksi, Respons, dan Pemulihan dengan tambahan studi literatur dan observasi. *Framework* ini dipilih karena cakupannya yang menyeluruh dibandingkan dengan *Cyber Kill Chain* dan *MITRE ATT&CK*.

3.1. Pengumpulan Data

Studi literatur digunakan untuk mengidentifikasi kesenjangan penelitian terkait integrasi *Wazuh*, *ModSecurity*, dan *Virus Total*. Observasi dilakukan secara tidak langsung melalui video, forum, dan artikel *online* untuk memahami konsep serta implementasi sistem keamanan yang diteliti.

3.2. Identifikasi

Tahap ini bertujuan untuk memetakan *asset*, topologi jaringan, dan kerentanan sistem. Identifikasi *asset* mencakup perangkat keras, perangkat lunak, dan kredensial untuk menentukan prioritas keamanan. topologi jaringan membantu memahami interaksi antar perangkat, sedangkan identifikasi kerentanan untuk mengetahui kerentanan yang akan diuji coba.

3.3. Proteksi

Proteksi dilakukan dengan konfigurasi *Wazuh* sebagai pusat *SIEM*, *ModSecurity* sebagai *WAF* yang menerapkan *OWASP Core Rule Set*, dan *Virus Total* sebagai layanan analisis *malware*. Integrasi ini memperkuat perlindungan terhadap ancaman siber melalui konfigurasi yang optimal.

3.4. Deteksi

Deteksi melibatkan integrasi sistem keamanan dan pengujian penetrasi berbasis *OWASP Top* 10. *ModSecurity* dikonfigurasi agar *log*nya dapat dianalisis oleh *Wazuh*, sementara *Wazuh Agent* memantau aktivitas sistem. Integrasi dengan *Virus Total* memungkinkan analisis *file* atau *URL* mencurigakan secara *real-time*. Pengujian penetrasi dilakukan untuk mengevaluasi efektivitas sistem terhadap ancaman.

3.5. Respon

Sistem keamanan ini menggunakan peringatan *real-time* dari *Wazuh*, pemblokiran otomatis oleh *ModSecurity*, dan analisis *file* dan *URL* dari *Virus Total*. Mekanisme otomatisasi memungkinkan respons cepat terhadap ancaman untuk meminimalkan dampak serangan.

3.6. Pemulihan

Pemulihan mencakup pemberian rekomendasi keamanan terhadap kerentanan tersebut, serta evaluasi dan penguatan kontrol akses untuk memastikan sistem tetap aman setelah insiden terjadi.

4. HASIL DAN PEMBAHASAN

4.1. Pengumpulan Data

Gambar 2. Dokumentasi *wazuh*

Hasil pengumpulan data menunjukkan kesenjangan dalam penelitian terdahulu yang membahas tentang *ModSecurity*, *Wazuh*, dan *Virus Total*. Penelitian ini bertujuan menjembatani kesenjangan tersebut dengan mengembangkan sistem deteksi dan analisis ancaman yang lebih efektif. Observasi melalui video, forum daring, dan artikel

online memberikan wawasan mendukung pengembangan sistem. Video membantu memahami penggunaan *tools*, forum memberikan pengalaman pengguna terkait implementasi dan strategi menghadapi ancaman, sementara artikel *online* menyediakan dokumentasi serta tahapan konfigurasi dan simulasi serangan seperti pada Gambar 2 Dokumentasi *wazuh*.

4.2. Identifikasi

Tahapan pertama dalam *NIST Framework* adalah identifikasi, yang bertujuan untuk memahami lingkungan sistem secara menyeluruh.

4.3. Identifikasi Kredensial

Tabel 1. Daftar kredensial

x	*Wazuh Server*	*Wazuh Dash board*	*Web Server*	*Local Web site*	*Attacker*
User name	ubuntu	admin	ubuntu	admin	kali
Pass word	ubuntu	admin	ubuntu	password	kali

Pada Tabel 1. Daftar kredensial, merupakan kredensial yang digunakan.

4.4. Identifikasi Perangkat Keras

Tabel 2. Daftar *hardware*

Komponen	**Spesifikasi**	
Jenis	***Laptop***	***Router***
Merk	*Lenovo Thinkpad T480*	*Huawei LTE Mobile Wifi*
CPU	*Intel Core I5-8350U*	-
RAM	*32GB*	-
Storage	*512GB + 128GB*	-
OS	*Windows 11 Pro*	-

Pada Tabel 2. Daftar Hardware, merupakan *hardware* yang digunakan.

4.5. Identifikasi Perangkat Lunak

Tabel 3. Daftar *software*

x	*Host*	*Wazuh Server*	*Web Server*	*Attacker*
Software	Windows 11	Ubuntu Server 24.04	Ubuntu Server 24.04	Kali Linux 2024.2
	Command Prompt	SSH Service	ModSecurity	Burp Suite
	Vmware Workstation	Wazuh Manager, Wazuh Indexer, Wazuh Dashboard	Apache Server	SQLMAP

Pada Tabel 3. Daftar *software*, merupakan *software* yang digunakan dalam penelitian ini.

4.6. Identifikasi Spesifikasi Virtual Machine

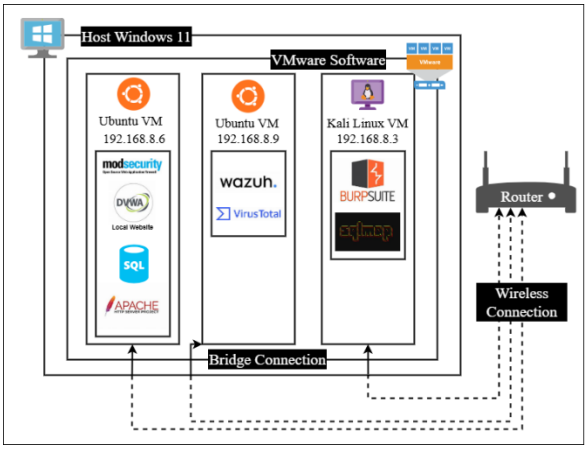
Tabel 4. Daftar spesifikasi *vm*

Komponen	***Wazuh***	***Web Server***	***Attacker***
Virtual Processor (core)	4	2	4
Virtual RAM	8 GB	4 GB	4 GB
Network Adapter	Bridge	Bridge	Bridge
Virtual Storage	124 GB	124 GB	80,1 GB

Pada Tabel 4. Daftar spesifikasi *vm*, merupakan spesifikasi dari *vm* yang digunakan dalam penelitian ini.

4.7. Identifikasi Topologi Jaringan

Pada Gambar 3. Topologi jaringan, terdapat terdapat gambaran dari topologi yang digunakan dalam penelitian ini.



Gambar 3. Topologi jaringan

4.8. Identifikasi Kerentanan

Tahap identifikasi bertujuan untuk memetakan celah keamanan dalam *local website* yang dapat dieksploitasi oleh penyerang. Penelitian ini mengacu pada *OWASP Top 10* tahun 2021 sebagai standar utama dalam mengidentifikasi dan memprioritaskan jenis kerentanan yang dapat dilihat pada Tabel 5. Daftar kerentanan.

Tabel 5. Daftar kerentanan

Category	***Title***	***Local Website Vunerability***
A01:2021	Broken Access Control	File Inclusion
		CSRF
		Weak Session IDs
		Authorization Bypass
A02:2021	Cryptographic Failures	Cryptographic
A03:2021	Injection	SQL Injection
		SQL Injection (Blind)
		Command Injection
		Reflected XSS
		Stored XSS
		DOM-Based XSS
		Open HTTP Redirect
A04:2021 & A06:2021	Insecure Design & Vulnerable and Outdated Components	Javascript
A05:2021	Security Misconfiguration	File Upload
A07:2021	Identification and Authentication Failures	Brute Force
		Insecure CAPTHA
A08:2021	Software and Data Integrity Failures	CSP Bypass

4.9. Proteksi

a. Installasi *Wazuh SIEM*

Penelitian ini dimulai dengan menetapkan alamat *IP* statis 192.168.8.9 untuk *Wazuh* melalui *50-cloud-init.yaml* di */etc/netplan*. *File* konfigurasi utama, seperti *wazuh-cert-tool.sh* dan *config.yml*, diunduh dan disesuaikan untuk mengatur *indexer*, *server*, dan *dashboard*.

Sertifikat keamanan dibuat menggunakan *wazuh-cert-tool.sh*, menghasilkan *file .pem* dan *.key* untuk komunikasi terenkripsi. Instalasi *Wazuh Indexer* dilakukan dengan *apt install wazuh-indexer*, diikuti konfigurasi *opensearch.yml* untuk pengaturan jaringan dan sertifikat.

Selanjutnya, *Wazuh Manager* dan *Filebeat* diinstal, dengan konfigurasi *ossec.conf* untuk aturan deteksi. *Filebeat* dikonfigurasi untuk koneksi jaringan dan penyimpanan data sensitif menggunakan *keystore*. Setelah layanan diuji dengan *filebeat test output*, *Wazuh Dashboard* dikonfigurasi dengan pengaturan *port*, alamat *IP*, dan *SSL*.

Setelah semua konfigurasi selesai, *dashboard Wazuh* berhasil diakses melalui *https*://192.168.8.9, menampilkan halaman *login* sebagai tanda keberhasilan instalasi seperti yang ditujukan pada Gambar 4. Tampilan *login wazuh*.

Gambar 4. Tampilan *login wazuh*

b. Installasi *Local Website*

Installasi dimulai dengan mengunduh *image Ubuntu Server* 24.04 dan mengubah konfigurasi *IP* dari *DHCP* ke statis untuk memastikan koneksi stabil.

Selanjutnya, *MySQL Server* diinstal dan diaktifkan dengan *service mysql start*, kemudian diverifikasi menggunakan *service mysql status*. *Database DVWA* dibuat, diikuti dengan pembuatan pengguna *user-dvwa* serta pemberian hak akses penuh.

Layanan *MySQL* dan *Apache2* kemudian dimulai ulang menggunakan *sudo service mysql restart* dan *sudo service apache2 restart*. Jika konfigurasi berhasil, halaman *login local website* dapat diakses sebagaimana ditunjukkan pada Gambar 5. Tampilan *login local website*.

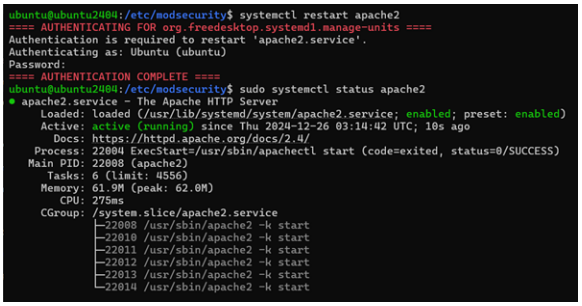
Gambar 5. Tampilan *login local website*

c. Installasi *ModSecurity*

Setelah *local website* terinstal, *ModSecurity* dipasang menggunakan perintah *sudo apt install libapache2-mod-security2*. Informasi paket *libapache 2-mod-security2* diperiksa dengan *apt-cache show*.

Untuk mencegah kesalahan konfigurasi, file konfigurasi di*backup* sebelum dimodifikasi. *SecRuleEngine* diubah dari *DetectionOnly* menjadi *On* agar sistem dapat memblokir serangan. Aturan *CRS* diunduh dengan *git clone* dan dikonfigurasi agar dikenali oleh *ModSecurity*, dan dapat meningkatkan perlindungan terhadap *local website*.

Sebagai langkah akhir, layanan *Apache2* dimulai ulang untuk menerapkan perubahan. Instalasi dan konfigurasi *ModSecurity* telah selesai, dan sistem berjalan tanpa kesalahan seperti pada Gambar 6. Melakukan *restart* terhadap *apache2*.



Gambar 6. Melakukan *restart* terhadap *apache2*

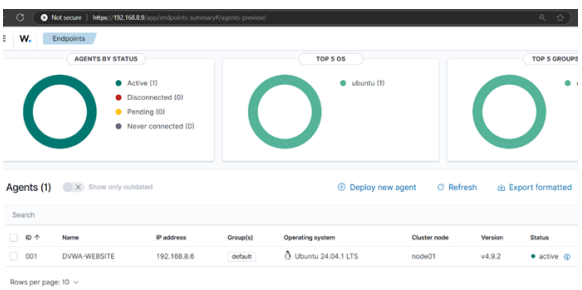
4.10. Deteksi

a. Integrasi *Endpoint* ke *Server*

Wazuh Agent diinstal pada *endpoint* untuk menghubungkannya dengan *server*. Proses dimulai dengan mengakses *Wazuh Dashboard*, memilih menu *Endpoint*, lalu opsi *Deploy New Agent*.

Pada menu instalasi, dipilih *Linux DEB amd64* sesuai dengan sistem operasi *Ubuntu Server*. Pengguna memasukkan alamat *IP server Wazuh* dan menentukan nama agen yang akan ditampilkan di *Dashboard Wazuh*, sementara konfigurasi grup dibiarkan *default*. Perintah instalasi yang tercantum dijalankan pada *endpoint* untuk menyelesaikan proses.

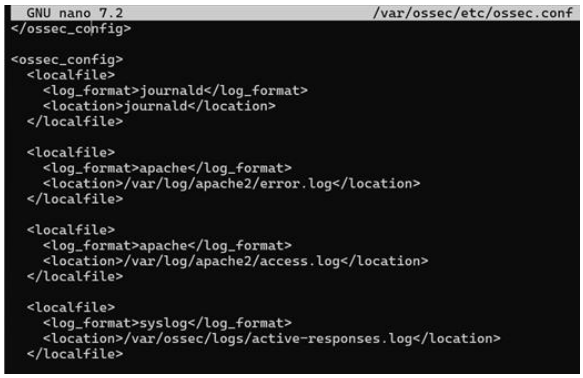
Setelah instalasi, *Wazuh Agent* berjalan dengan *status active* di *endpoint*, yang dapat diverifikasi melalui *Dashboard Wazuh* seperti pada Gambar 7. Tampilan agen yang aktif. Agen ini berfungsi untuk memantau aktivitas sistem secara *real-time*, mendeteksi ancaman, dan mengirimkan *log* ke *server* untuk analisis lebih lanjut.



Gambar 7. Tampilan agen yang aktif

b. Integrasi *Modsecurity* ke *Wazuh*

Tahapan integrasi *ModSecurity* dengan *Wazuh* memiliki satu tahapan, yaitu penambahan *file error.log* ke dalam konfigurasi *Wazuh Agent* yang ditunjukan pada Gambar 8. Tampilan konfigurasi wazuh agent.



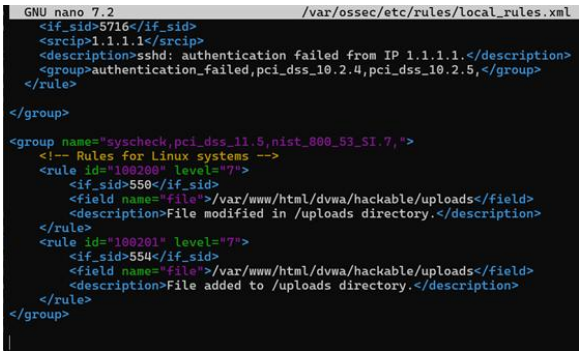
Gambar 8. Tampilan konfigurasi wazuh agent

Dengan demikian, penulis dapat memantau aktivitas pencegahan serangan *website* secara terpusat melalui *dashboard wazuh*.

c. Integrasi *Virus Total* dengan *Wazuh*

Integrasi dimulai dengan membuat akun *Virus Total* untuk memperoleh *API Key*, yang digunakan sebagai kredensial utama dalam menghubungkan *Virus Total* dengan *Wazuh Server*. *API Key* ini dimasukkan ke dalam *file* konfigurasi *ossec.conf* agar *server Wazuh* dapat berkomunikasi dengan *Virus Total*.

Untuk mendeteksi *file* berbahaya, *rules* khusus ditambahkan seperti pada Gambar 9. Pembuatan *rules virus total*, sehingga ketika *file malicious* ditemukan di direktori *uploads*, sistem akan menghasilkan *event* pada *Wazuh Dashboard*. Selanjutnya, skrip *remove-threat.sh* dikonfigurasi pada *endpoint* untuk menghapus *file* berbahaya secara otomatis.



Gambar 9. Pembuatan *rules virus total*

Skrip ini diintegrasikan dengan *Wazuh Agent*, dan izin aksesnya diatur sebelum melakukan *restart Wazuh Agent*. Dengan integrasi ini, *Wazuh* dapat memonitor *file* berbahaya pada direktori *uploads* melalui analisis *log event* di *Dashboard Wazuh*.

d. *Website Penteration Testing*

Eksperimen dilakukan dalam lingkungan *virtual* untuk menguji kemampuan sistem dalam mendeteksi

berbagai jenis serangan siber berdasarkan *OWASP Top* 10 tahun 2021. Pengujian meliputi *File Inclusion Attack*, baik *Local* maupun *Remote*, untuk mengakses *file* sistem atau menanamkan *reverse shell*. *CSRF* (*Cross-Site Request Forgery*) diuji melalui serangan *Man-in-the-Middle* (*MITM*) dengan *brute force*. *Weak Session IDs* dieksploitasi melalui pencurian *cookie*, sementara *Authorization Bypass* memungkinkan pengguna biasa mengakses menu *administrator*. *Cryptographic Failures* diuji tetapi sistem tidak mampu mendeteksi kelemahan kriptografi. Serangan *SQL Injection* dan *Blind SQL Injection* mencakup manipulasi *query* untuk mengakses atau mengekstrak data dari basis data secara langsung maupun tersembunyi. *Command Injection* memungkinkan eksekusi perintah sistem melalui input berbahaya. Tiga jenis *XSS* (*Cross-Site Scripting*), yaitu *Reflected*, *Stored*, dan *DOM-Based*, disimulasikan melalui skrip yang disisipkan ke dalam halaman *website*. *Open HTTP Redirect* memungkinkan pengalihan ke *URL* berbahaya, sementara *File Upload* menguji kemampuan sistem dalam mengenali *file* berbahaya seperti *EICAR*. Uji lainnya mencakup manipulasi skrip *JavaScript*, serangan *Brute Force* menggunakan *rainbow attack*, penyalahgunaan *Insecure CAPTCHA* dengan respons yang dapat digunakan ulang, dan *CSP Bypass* dengan *URL* yang mengandung skrip berbahaya.

Tabel 6. Hasil percobaan penyerangan

Jenis Serangan	Hasil Pengujian	Pencegahan
File Inclusion Attack	Berhasil terdeteksi dan diblokir	Menggunakan *ModSecurity* pada *Apache2*, menerapkan validasi *input* berbasis *whitelisting*, serta penggunaan *absolute path* untuk mencegah serangan.
CSRF (Cross-Site Request Forgery)	serangan tidak terdeteksi. Perubahan nilai parameter *password* tidak dipantau.	Konfigurasi *ModSecurity* untuk memblokir permintaan *POST* tanpa token *CSRF*. Gunakan metode *POST* untuk perubahan *password*. Terapkan *Anti-CSRF* token dan *hashing password* dengan *bcrypt*.
Weak Session IDs	serangan tidak terdeteksi, penyerang berhasil mencuri *cookie administrator*	Perlu penerapan *session ID* yang lebih acak dan kuat (misal: *bin2hex(random_bytes(32*))), serta konfigurasi *cookie* dengan atribut *Secure*, *HttpOnly*, dan *SameSite*. *ModSecurity* dapat digunakan untuk memblokir *session* ID lemah, dan *Wazuh* untuk memonitor pola *session* ID yang tidak aman.
Authorization Bypass	serangan tidak terdeteksi, tetapi perlu analisis lebih dalam dan permintaan tidak diblokir.	Perlu penambahan aturan pada *ModSecurity* dan implementasi kode tambahan untuk membatasi akses hanya bagi pengguna dengan hak *administrator*.
Cryptography	serangan tidak terdeteksi.	Implementasi algoritma kurang aman. Hindari *XOR + Base64*, gunakan *AES*/*RSA* dengan kunci unik per sesi/data. Terapkan enkripsi *end-to-end* untuk keamanan data.
SQL Injection	Berhasil terdeteksi dan diblokir	*SQL Injection* dapat dicegah dengan *prepared statements* untuk memisahkan kode *SQL* dari *input*, validasi dan *escaping input* untuk menghindari eksekusi perintah berbahaya, serta pembatasan hak akses guna mencegah eksploitasi. Selain itu, *ModSecurity* efektif dalam mendeteksi dan memblokir serangan dengan memantau log aktivitas.
SQL Injection (Blind)	Berhasil terdeteksi dan diblokir	Pencegahan dilakukan dengan *prepared statements* atau *parameterized queries* untuk memisahkan perintah *SQL* dari input pengguna. *ModSecurity* juga berperan penting dalam mendeteksi dan mencegah serangan.
Command Injection	Berhasil terdeteksi dan diblokir	Pencegahan dilakukan dengan validasi input menggunakan *filter_var*(), membatasi fungsi seperti *shell_exec*(), serta menerapkan *escapeshellarg*(). Selain itu, membatasi hak akses server dan menggunakan *ModSecurity* untuk deteksi serta pencegahan serangan.
Reflected XSS	Berhasil terdeteksi dan diblokir	Menggunakan *ModSecurity* untuk memblokir *payload* berbahaya dan menerapkan sanitasi input menggunakan fungsi seperti *htmlspecialchars*().
Stored XSS	Berhasil terdeteksi dan diblokir	Hindari menyimpan data mentah dari *input* pengguna tanpa *filtering*. Terapkan *escaping*, encoding aman, dan *modsecurity* dengan *Core Rule Set* untuk cegah *Stored XSS*.
DOM-Based XSS	Berhasil terdeteksi dan diblokir	hindari manipulasi langsung *DOM* menggunakan data dari *URL* atau *input* tanpa validasi. Gunakan *API* aman seperti *textContent* atau *innerText*, bukan *innerHTML*, dan terapkan *ModSecurity*.
Open HTTP Redirect	Berhasil terdeteksi dan diblokir	validasi dengan ketat *URL* tujuan, gunakan whitelist *domain*, dan lakukan *tuning ModSecurity* untuk blokir permintaan mencurigakan.

Jenis Serangan	Hasil Pengujian	Pencegahan
File Upload	*ModSecurity* berhasil memblokir unggahan file pada skenario pertama. Pada skenario kedua, file berhasil diunggah tetapi diidentifikasi sebagai *malicious* oleh *VirusTotal* dan dihapus otomatis oleh *Wazuh*.	Pencegahan kerentanan *file upload* dalam *local website* meliputi validasi sisi server dan *klien*, batasi jenis dan ukuran file, serta simpan file di direktori terisolasi tanpa izin eksekusi.
JavaScript	serangan tidak terdeteksi, dan perlu analisis lebih dalam dan permintaan tidak diblokir.	terapkan validasi *input* di server dan klien, hindari fungsi *hashing* lemah seperti *MD5*, dan lakukan *tuning rules Wazuh* dan *ModSecurity*.
Brute Force	Berhasil terdeteksi dan diblokir	Melakukan pembatasan percobaan *login based on* alamat ip dan menerapkan blokir alamat ip selama waktu tertentu.
Insecure CAPTCHA	serangan tidak terdeteksi, dan perlu analisis lebih dalam dan permintaan tidak diblokir.	Evaluasi pencegahan kerentanan *CAPTCHA* dilakukan dengan validasi di setiap tahap, penggunaan token sesi unik, serta tuning *ModSecurity* untuk mendeteksi dan memblokir serangan.
CSP Bypass	serangan tidak terdeteksi, dan perlu analisis lebih dalam dan permintaan tidak diblokir.	validasi *input* ketat untuk terima *URL* terpercaya, tambahkan *hash* pada *script* yang diizinkan untuk cegah eksekusi *script* berbahaya.

Dari hasil pengujian terhadap berbagai jenis serangan siber, sebagian besar serangan yang termasuk dalam kategori *OWASP Top* 10 berhasil dideteksi dan diblokir, khususnya serangan *SQL Injection*, *Command Injection*, berbagai varian *XSS* (*Reflected*, *Stored*, *DOM-Based*), *File Inclusion*, *Open HTTP Redirect, dan Brute Force*. Keberhasilan ini didukung oleh penerapan *ModSecurity*, validasi *input*, *prepared statements*, dan mekanisme pembatasan akses. Namun demikian, terdapat beberapa serangan yang tidak terdeteksi, seperti *CSRF*, *Weak Session IDs*, *Authorization Bypass*, *Cryptography Flaws*, *JavaScript Manipulation*, *Insecure CAPTCHA*, dan *CSP Bypass*. Kelemahan ini menunjukkan perlunya peningkatan dalam penerapan token keamanan, penguatan sesi, validasi *input* yang lebih ketat, serta *tuning* aturan *ModSecurity* dan *Wazuh*.

5. KESIMPULAN DAN SARAN

Penelitian ini berhasil mengintegrasikan *Wazuh SIEM*, *ModSecurity WAF*, dan *VirusTotal* ke dalam satu sistem keamanan terpadu yang telah diuji dan terbukti mampu meningkatkan deteksi ancaman pada *website*. Untuk mengoptimalkan efektivitas sistem, diperlukan penyesuaian aturan pada *Wazuh* dan *ModSecurity*, serta penerapan mekanisme tambahan seperti *Anti-CSRF token*, *session ID* yang lebih aman, pembatasan akses berbasis peran, enkripsi kuat (*AES* atau *RSA*), validasi input yang ketat, dan konfigurasi *Content Security Policy* (*CSP*) yang lebih baik. Penelitian selanjutnya disarankan untuk melakukan *tuning* lanjutan pada aturan keamanan serta mengintegrasikan sistem ini dengan teknologi keamanan lainnya guna memastikan deteksi dan mitigasi yang lebih komprehensif terhadap seluruh kerentanan dalam *OWASP Top* 10 Tahun 2021.

DAFTAR PUSTAKA

[1] Badan Siber dan Sandi Negara (BSSN). 2023. Lanskap Keamanan Siber Indonesia 2023. Jakarta, Indonesia: BSSN.

[2] Harefa, J., Prajena, G., Alexander, A., Muhamad, A., Dewa, E. V. S., & Yuliandry, S. 2021. *Sea waf: the prevention of sql injection attacks on web applications. Advances in Science, Technology and Engineering Systems Journal*, 6(2), 405-411. https://doi.org/10.25046/aj060247

[3] Badan Siber dan Sandi Negara, Id-SIRTII/CC. 2023. Laporan Bulanan Publik Hasil Monitoring Keamanan Siber 2023.

[4] *IBM*,"*What is Security Information and Event Management SIEM*? " *IBM*, 2022, [2 November 2024]. https://www.ibm.com/id-en/topics/siem

[5] Linode. (2021). *Securing Apache2 with ModSecurity*. https://www.linode.com/docs/guides/securing-apache2-with-modsecurity/

[6] Choo, E, Nabeel, M, Kim, D, Silva, R De, Yu, T, & ... 2024. *A large scale study and classification of virustotal reports on phishing and malware urls. ACM SIGMETRICS* …, dl.acm.org, https://doi.org/10.1145/3673660.3655042

[7] Ayunda, K. D., Widjajarto, A., & Budiono, A. 2021. Implementation and *Analysis ModSecurity* on *Web-Based Application with OWASP Standards*. Jurnal Teknik Informatika dan Sistem Informasi, 8(3), 1638-1650. Universitas Telkom.

[8] AZZAH, S. 2024. Implementasi Sistem Keamanan Jaringan Di Psdku Universitas Lampung Waykanan Menggunakan *Server Wazuh* Untuk Deteksi Dan Respon Serangan Siber., http://digilib.unila.ac.id/id/eprint/78645

[9] Khan, Wasif. 2022. *Achieving Regulatory Compliance with ISO 27001 and NIST Frameworks: The Process and Challenges of Obtaining these Critical Certifications for Clients. Journal of Artificial Intelligence & Cloud Computing.* 1-14. 10.47363/JAICC/2022(1)E170.

[10] Vielberth, M. 2021. *Security information and event management (SIEM). Encyclopedia of Cryptography, Security and Privacy, Springer*,

https://doi.org/10.1007/978-3-642-27739-9_1681-1

[11] Aditya, R, Muhyidin, Y, & Singasatia, D. 2024. Implementasi *Security Information and Event Management* (*SIEM*) Untuk *Monitoring* Keamanan *Server* Menggunakan *Wazuh*. Merkurius: Jurnal Riset …, journal.arteii.or.id, https://journal.arteii.or.id/index.php/Merkurius/article/view/289

[12] *Cisco*. (n.d.). *What is a Web Application Firewall (WAF)? Cisco*. *Retrieved* [2 November 2024], https://www.cisco.com/site/us/en/learn/topics/security/what-is-web application-firewall-waf.html

[13] Virdi, S. (n.d.). *Web and Data Processing, Chapter 1. Kent State University. Retrieved* [2 November 2024], https://www.cs.kent.edu/~svirdi/Ebook/wdp/ch01.pdf

[14] *Wazuh. (n.d.). Wazuh - Open Source Security Platform. Retrieved* [2 November 2024], from https://wazuh.com/

[15] *ModSecurity. (n.d.). ModSecurity: Open Source Web Application Firewall (WAF). Retrieved* [2 November 2024], from https://modsecurity.org/

[16] Salem, A., Banescu, S., & Pretschner, A. (2020). *Maat: Automatically Analyzing VirusTotal for Accurate Labeling and Effective Malware Detection*.

[17] *Scarfone, K., & Grance, T. (2022). Framework for Improving Critical Infrastructure Cybersecurity: Version 1.1 (NIST Special Publication 1271). National Institute of Standards and Technology. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1271.pdf*

[18] Manico, J. (2022). *OWASP Top Ten 2021: The Ten Most Critical Web Application Security Risks*. SecAppDev. https://handouts.secappdev.org/handouts/2022/jimmanico_owasptop10.pdf

[19] Ardiansyah, Y., Sunandar, M. A., & Muhyidin, Y. (2023). *Implementasi* keamanan *website* dengan metode *firewall* aplikasi *web (WAF):* Studi kasus *WEB* Desa Wantilan. JATI (Jurnal Mahasiswa Teknik Informatika), 7(3), 2018-2025. https://mail.ejournal.itn.ac.id/index.php/jati/article/view/7048/4222

[20] Aditya, R, Muhyidin, Y, & Singasatia, D. 2024. Implementasi *Security Information and Event* Management (SIEM) Untuk *Monitoring* Keamanan *Server* Menggunakan *Wazuh*. Merkurius: Jurnal Riset …, journal.arteii.or.id, https://journal.arteii.or.id/index.php/Merkurius/article/view/289

[21] Kantola, T. (2022). *Exploring VirusTotal for security operations alert triage automation* (Bachelor's thesis). JAMK *University* of *Applied Sciences*. *Retrieved* from https://www.theseus.fi/bitstream/handle/10024/704502/Thesis_Kantola_Teemu.pdf?sequence=2&isAllowed=y