

Deteksi dan Mitigasi Serangan Aplikasi Web Menggunakan *Suricata IPS*, *Wazuh SIEM*, dan *Firewall*

Ardiansyah^{1*}, Raisul Azhar¹, Lilik Widyawati¹

¹ Program Studi Ilmu Komputer, Universitas Bumigora, Indonesia

* Corresponding author: ardiansyah240202@gmail.com

Sitasi: A. Ardiansyah, R. Azhar, and L. Widyawati, "Deteksi dan Mitigasi Serangan Aplikasi Web Menggunakan *Suricata IPS*, *Wazuh SIEM*, dan *Firewall*," *Jurnal Teknologi Informasi dan Multimedia*, vol. 8, no. 4, pp. 999–1020, 2026.

Diterima : 01-08-2026
Direvisi : 21-09-2026
Disetujui : 05-10-2026
Diterbitkan : 08-10-2026

Copyright: © 2026 oleh para penulis. Karya ini dilisensikan di bawah Creative Commons Attribution-Share Alike 4.0 International License.

Abstrak. Aplikasi *web* telah menjadi *platform* utama dalam menyediakan layanan digital, tetapi menghadapi ancaman keamanan, khususnya *SQL Injection* (SQLi) dan *Cross-Site Scripting* (XSS). Keterbatasan *Firewall* konvensional dalam mendeteksi serangan pada lapisan aplikasi mendorong perlunya sistem dengan kemampuan deteksi dan mitigasi otomatis. Penelitian ini mengevaluasi kerangka kerja keamanan terintegrasi berbasis *Suricata Intrusion Prevention System* (IPS), *Wazuh Security Information and Event Management* (SIEM), dan *Firewall* pada lingkungan *cloud*. Metode *Network Development Life Cycle* (NDLC) diterapkan pada *Ubuntu Server* dan *Damn Vulnerable Web Application* (DVWA). Pengujian dilakukan pada empat skenario: *Error-Based SQLi*, *Union-Based SQLi*, *Authentication Bypass SQLi*, dan XSS. Hasil menunjukkan bahwa tiga skenario (75%) berhasil dideteksi, direspons, dan diblokir melalui *Active Response*. *Suricata* mencatat tujuh *event* deteksi pada *Error-Based SQLi* serta masing-masing empat *alert* pada *Union-Based SQLi* dan XSS. *Wazuh* mencatat *alert* dan menjalankan *Active Response* pada ketiga skenario tersebut, sedangkan *Firewall* memblokir masing-masing 281, 215, dan 73 paket. *Authentication Bypass SQLi* menghasilkan 46 *event* *Suricata* dan 15 *alert* *Wazuh*, tetapi tidak memicu *Active Response* maupun pemblokiran *Firewall* karena *SQLMap* tidak mengidentifikasi parameter *username* sebagai rentan. Hasil ini menunjukkan bahwa integrasi *Suricata IPS*, *Wazuh SIEM*, dan *Firewall* mendukung deteksi, analisis, dan mitigasi serangan aplikasi *web* secara otomatis melalui pendekatan *defense in depth*.

Kata kunci: *Suricata IPS*, *Wazuh SIEM*, *Firewall*, *SQLi*, *Stored XSS*

Abstract. Web applications have become a primary platform for delivering digital services but face security threats, particularly *SQL Injection* (SQLi) and *Cross-Site Scripting* (XSS). The limitations of conventional firewalls in detecting application-layer attacks highlight the need for automated detection and mitigation. This study evaluates an integrated security framework based on *Suricata Intrusion Prevention System* (IPS), *Wazuh Security Information and Event Management* (SIEM), and a firewall in a cloud environment. The *Network Development Life Cycle* (NDLC) method was implemented on *Ubuntu Server* and *Damn Vulnerable Web Application* (DVWA). Four attack scenarios were tested: *Error-Based SQLi*, *Union-Based SQLi*, *Authentication Bypass SQLi*, and XSS. Results showed that three scenarios (75%) were successfully detected, responded to, and blocked through *Active Response*. *Suricata* recorded seven detection events for *Error-Based SQLi* and four alerts each for *Union-Based SQLi* and XSS. *Wazuh* recorded alerts and executed *Active Response* for these scenarios, while the firewall blocked 281, 215, and 73 packets, respectively. *Authentication Bypass SQLi* generated 46 *Suricata* events and 15 *Wazuh* alerts but did not trigger *Active Response* or firewall blocking because *SQLMap* did not identify the *username* parameter as vulnerable. These findings demonstrate that integrating *Suricata IPS*, *Wazuh SIEM*, and a firewall supports auto-mated detection, analysis, and mitigation of web attacks through a *defense-in-depth* approach.

Keywords: *Suricata IPS*, *Wazuh SIEM*, *Firewall*, *SQLi*, *Stored XSS*

1. Pendahuluan

Kemajuan teknologi informasi telah mempercepat adopsi aplikasi berbasis *web* sebagai *platform* utama dalam menyediakan layanan digital di berbagai sektor. Aksesibilitas, fleksibilitas, dan efisiensinya telah berkontribusi terhadap penggunaannya secara luas dalam pengelolaan informasi dan layanan berbasis Internet [1]. Namun demikian, meningkatnya ketergantungan pada aplikasi *web* juga meningkatkan kekhawatiran terhadap keamanan, karena aplikasi tersebut telah menjadi target utama serangan siber yang dapat mengakibatkan kebocoran data, manipulasi informasi, dan gangguan layanan [2, 3].

Di antara ancaman keamanan yang umum menargetkan aplikasi *web* adalah SQLi dan Stored XSS. Berdasarkan data laporan serangan di Indonesia, SQLi tercatat sebanyak 79 laporan, sedangkan Stored XSS sebanyak 46 laporan pada tahun 2022 [4]. Ancaman tersebut juga masih terlihat dalam perkembangan keamanan aplikasi *web* pada tahun-tahun berikutnya. Data keamanan WordPress [5] mencatat sekitar 9 miliar percobaan eksploitasi Stored XSS dan 1,1 miliar percobaan SQLi yang berhasil diblokir selama tahun 2024. Data tersebut menunjukkan bahwa kedua jenis serangan masih menjadi bagian signifikan dari aktivitas eksploitasi terhadap aplikasi *web*. SQLi mengeksploitasi kelemahan dalam validasi masukan dengan menyisipkan pernyataan SQLi berbahaya ke dalam aplikasi, yang dapat memungkinkan akses tidak sah ke basis data, pengambilan informasi sensitif, atau modifikasi data [4]. Sementara itu, Stored XSS dilakukan dengan menyisipkan skrip berbahaya ke dalam aplikasi *web* sehingga dapat dieksekusi pada peramban pengguna lain dan berpotensi menyebabkan pencurian informasi atau manipulasi konten secara tidak sah [3]. Kondisi tersebut menunjukkan pentingnya penerapan mekanisme keamanan yang mampu mendeteksi dan memitigasi kedua jenis serangan tersebut secara efektif [6].

Firewall merupakan salah satu mekanisme keamanan yang paling banyak diterapkan untuk melindungi infrastruktur jaringan. *Firewall* mengatur akses jaringan dengan melakukan penyaringan terhadap lalu lintas masuk dan keluar berdasarkan alamat IP, nomor port, dan protokol komunikasi [7]. Meskipun efektif pada lapisan jaringan, *Firewall* memiliki keterbatasan karena tidak dapat melakukan pemeriksaan mendalam terhadap isi paket untuk mengidentifikasi pola serangan kompleks yang terjadi pada lapisan aplikasi [8]. Oleh karena itu, diperlukan teknologi keamanan tambahan, seperti IDS dan IPS, untuk meningkatkan kemampuan deteksi dan pencegahan terhadap ancaman [9].

Suricata merupakan solusi keamanan jaringan *open-source* yang dapat beroperasi sebagai IDS maupun IPS. Suricata melakukan inspeksi lalu lintas secara *real-time* dengan menerapkan aturan deteksi berbasis *signature* untuk mengidentifikasi aktivitas berbahaya dan menghasilkan *log* keamanan untuk analisis lebih lanjut [10]. Penelitian sebelumnya telah menunjukkan bahwa Suricata dapat diterapkan secara efektif sebagai IPS untuk mendeteksi serangan SQLi melalui aturan deteksi yang dikonfigurasi dengan tepat [11]. Namun, penggunaan IPS secara mandiri menimbulkan tantangan dalam pengelolaan *log* secara terpusat dan analisis kejadian keamanan secara menyeluruh, khususnya pada lingkungan dengan volume aktivitas jaringan yang tinggi [12].

Keterbatasan tersebut dapat diatasi melalui penerapan *platform* Wazuh SIEM, yang menyediakan pengumpulan *log* secara terpusat, korelasi kejadian, dan analisis keamanan [13]. Wazuh merupakan solusi SIEM *open-source* yang menyediakan pemantauan keamanan secara berkelanjutan, analisis *log*, dan respons insiden otomatis melalui kemampuan *Active Response* [14]. Integrasi Suricata dengan Wazuh memungkinkan *log* keamanan yang dihasilkan dan dianalisis secara terpusat, sehingga meningkatkan efektivitas proses identifikasi ancaman dan mitigasi otomatis [15].

Evaluasi mekanisme keamanan aplikasi *web* memerlukan lingkungan pengujian yang terkontrol untuk memastikan bahwa eksperimen dapat dilakukan tanpa

memengaruhi sistem produksi. Dalam penelitian ini, DVWA digunakan sebagai aplikasi target untuk pengujian keamanan[16]. DVWA dirancang secara khusus dengan berbagai kerentanan keamanan, termasuk SQLi dan *Stored XSS*, sehingga menjadi *platform* yang banyak digunakan untuk eksperimen keamanan aplikasi *web*[4]. Penggunaannya memfasilitasi pelaksanaan berbagai skenario serangan untuk menilai efektivitas sistem keamanan yang diusulkan dalam mendeteksi dan memitigasi potensi ancaman[16].

Beberapa penelitian sebelumnya telah mengkaji penerapan Suricata dan Wazuh untuk meningkatkan keamanan jaringan. Elsa Stephani[10] dan Anugrah et al.[11] menunjukkan bahwa Suricata mampu mendeteksi serangan SQLi melalui penerapan aturan deteksi yang disesuaikan. Demikian pula, Setiawan et al.[17] melaporkan bahwa integrasi Suricata dengan *platform* SIEM meningkatkan pemantauan keamanan dan analisis kejadian dengan memungkinkan pengelolaan *log* secara terpusat. Namun, penelitian tersebut terutama menekankan kemampuan deteksi dan pemantauan, sedangkan integrasi antara deteksi serangan, analisis kejadian keamanan, dan mitigasi otomatis melalui mekanisme *Firewall* masih belum banyak dieksplorasi.

Berdasarkan kesenjangan penelitian tersebut, penelitian ini mengusulkan implementasi kerangka kerja keamanan aplikasi *web* dengan mengintegrasikan Suricata sebagai IPS, Wazuh sebagai *platform* SIEM, dan *Firewall* sebagai mekanisme mitigasi otomatis. Kerangka kerja yang diusulkan dievaluasi menggunakan DVWA dengan skenario serangan SQLi dan *Stored XSS* untuk menilai kemampuannya dalam mendeteksi, menganalisis, dan merespons serangan siber. Pendekatan terintegrasi ini diharapkan dapat memberikan arsitektur keamanan berlapis yang lebih komprehensif melalui penggabungan fungsi deteksi, pemantauan, dan mitigasi otomatis, sehingga menawarkan efektivitas yang lebih baik dibandingkan dengan penerapan masing-masing mekanisme keamanan secara terpisah.

2. Bahan dan Metode

2.1. Bahan

Penelitian ini menggunakan tiga *virtual machine* (VM) yang dikonfigurasi dalam jaringan *virtual* untuk membangun lingkungan pengujian terisolasi. Spesifikasi perangkat keras utama yang digunakan adalah prosesor Intel Core i5, RAM 12 GB, dan SSD 256 GB. Konfigurasi perangkat lunak dan fungsi masing-masing VM ditunjukkan pada Tabel 1.

Tabel 1. Konfigurasi perangkat dan perangkat lunak penelitian

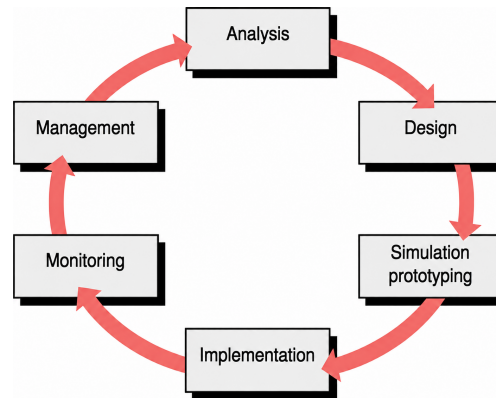
VM	Sistem Operasi	Perangkat Lunak	Fungsi
Kali Linux	Kali Linux	SQLMap	Sumber serangan dan pengujian SQLi dan <i>Stored XSS</i>
Cloud Server/DVWA	Ubuntu 24.04.4 LTS	Server	Target pengujian, deteksi, dan mitigasi serangan
Wazuh SIEM	Ubuntu 24.04.3 LTS	Server	Pengumpulan, analisis, dan visualisasi <i>log</i> keamanan
		DVWA, Suricata 8.0.4, Wazuh Agent, Firewall, Wazuh Manager, Wazuh Indexer, Wazuh Dashboard 4.14.5	

Kali Linux digunakan sebagai mesin penyerang, sedangkan *Ubuntu Server* berfungsi sebagai *web server* yang menjalankan DVWA sebagai target pengujian sekaligus Suricata, Wazuh Agent, dan *Firewall*. *Ubuntu Server* lainnya digunakan sebagai *server* SIEM untuk menjalankan Wazuh Manager, Indexer, dan Dashboard.

2.2. Metode Penelitian

Penelitian ini mengadopsi metodologi *Network Development Life Cycle* (NDLC) sebagai kerangka kerja dalam perancangan dan implementasi sistem keamanan

jaringan. NDLC dipilih karena menyediakan pendekatan terstruktur dalam pengembangan infrastruktur jaringan melalui tahapan *Analysis*, *Design*, *Simulation Prototyping*, dan *Implementation* [18]. Pada tahap *Analysis*, dilakukan identifikasi kebutuhan sistem keamanan serta karakteristik serangan yang akan diuji. Tahap *Design* mencakup perancangan arsitektur integrasi Suricata IPS, Wazuh SIEM, dan *Firewall* sebagai sistem keamanan terintegrasi. Selanjutnya, tahap *Simulation Prototyping* mencakup pembangunan dan pengujian rancangan pada lingkungan *virtual* sebelum diterapkan pada skenario pengujian.



Gambar 1. Tahapan pada NDLC [19]

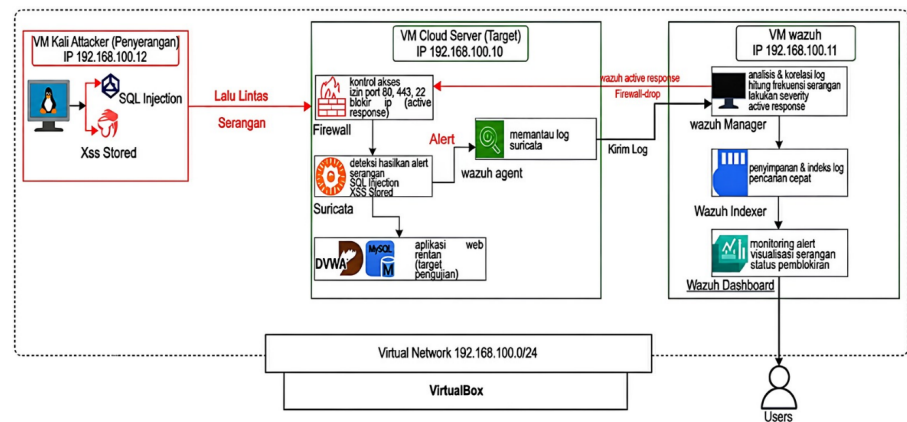
Terakhir, tahap *Implementation* mencakup konfigurasi dan integrasi seluruh komponen keamanan untuk mendeteksi, memantau, dan memitigasi serangan pada aplikasi *web* DVWA. Alur penerapan keempat tahapan NDLC dalam penelitian ini ditunjukkan pada Gambar 1.

2.2.1. Tahap Analysis

Tahap *Analysis* dilakukan melalui studi literatur dengan mengkaji artikel ilmiah, jurnal, skripsi, dan publikasi yang relevan dengan keamanan aplikasi *web*. Pada tahap ini, skenario serangan yang akan diuji diidentifikasi, meliputi *Error-Based SQLi*, *Union-Based SQLi*, *Authentication Bypass SQLi*, dan *Stored XSS*. Selain itu, dilakukan analisis kebutuhan perangkat keras, perangkat lunak, dan tools pendukung untuk membangun sistem keamanan terintegrasi yang terdiri atas Suricata IPS, Wazuh SIEM, dan *Firewall*.

2.2.2. Tahap Design

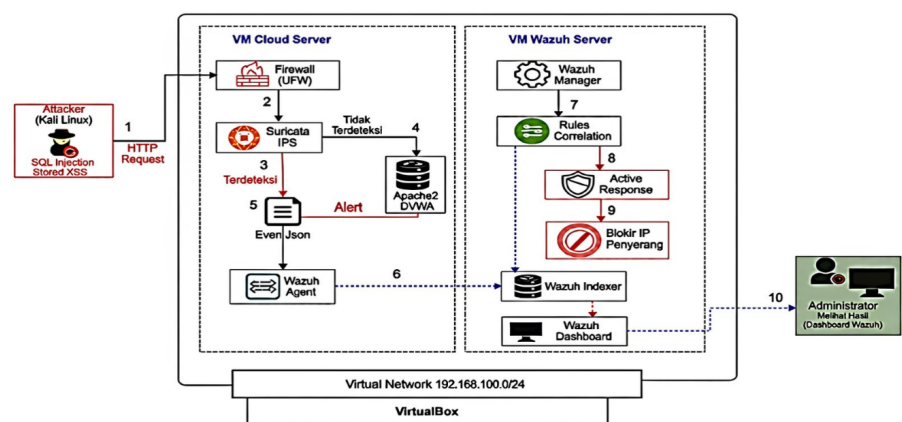
Tahap *Design* berfokus pada perancangan arsitektur keamanan dalam lingkungan virtualisasi yang terdiri atas tiga mesin *virtual*, yaitu mesin penyerang, *server* aplikasi *web*, dan *server* SIEM. Ketiga mesin tersebut dirancang untuk menjalankan fungsi simulasi serangan, penyediaan layanan aplikasi *web*, serta pemantauan dan analisis keamanan. Lalu lintas menuju DVWA diperiksa oleh Suricata IPS menggunakan aturan deteksi custom. Hasil deteksi kemudian dicatat dalam *log* dan diteruskan ke Wazuh untuk analisis serta korelasi *event*. Ketika kondisi *rule* mitigasi terpenuhi, Wazuh *Active Response* menjalankan *Firewall-drop* untuk memblokir alamat IP sumber melalui *Firewall*. Rancangan jaringan penelitian ditunjukkan pada Gambar 2.



Gambar 2. Rancangan Topologi Jaringan

Gambar 2 menunjukkan Topologi jaringan penelitian yang diterapkan menggunakan *VirtualBox* pada jaringan *virtual* 192.168.100.0/24. Topologi terdiri atas *Kali Attacker* (192.168.100.12) sebagai sumber *simulasi* serangan, *Cloud Server* (192.168.100.10) sebagai *server* target yang menjalankan DVWA, Suricata IPS, Firewall, dan Wazuh Agent, serta Wazuh SIEM (192.168.100.11) yang menjalankan Wazuh Manager, Wazuh Indexer, dan Wazuh Dashboard. Selama pengujian, lalu lintas dari *Kali Attacker* menuju DVWA diperiksa oleh Suricata berdasarkan aturan deteksi yang dikonfigurasi. Alert yang dihasilkan diteruskan melalui Wazuh Agent ke Wazuh Manager untuk dianalisis dan dikorelasikan. Ketika kondisi *rule* mitigasi terpenuhi, Wazuh Active Response menjalankan Firewall-drop untuk memblokir alamat IP sumber. Aktivitas deteksi dan mitigasi selanjutnya dipantau melalui Wazuh Dashboard.

Rancangan keamanan penelitian mengintegrasikan Suricata sebagai IPS, Wazuh SIEM, dan Firewall sebagai mekanisme mitigasi untuk mendukung proses deteksi hingga pemblokiran serangan secara otomatis, sebagaimana ditunjukkan pada Gambar 3.

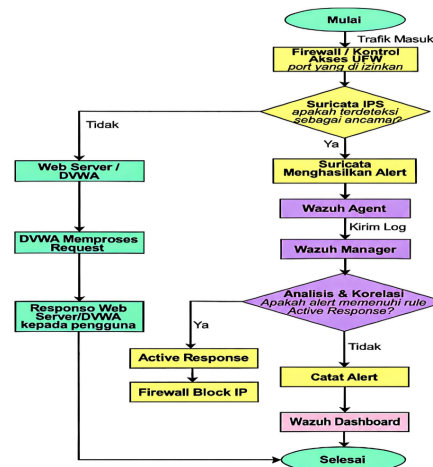


Gambar 3. Rancangan Alur Kerja Sistem Keamanan

Arsitektur keamanan mendukung inspeksi trafik secara *real-time*, analisis kejadian keamanan, dan respons otomatis terhadap serangan. Alur dimulai ketika mesin penyerang mengirimkan trafik HTTP menuju *server* aplikasi *web* melalui Firewall, kemudian trafik diperiksa oleh Suricata IPS menggunakan aturan custom. Jika trafik tidak terindikasi sebagai serangan, *request* diteruskan ke DVWA untuk diproses dan menghasilkan HTTP *response* tanpa memicu Alert. Sebaliknya, apabila trafik terdeteksi mengandung pola SQLi atau Stored XSS, Suricata melakukan tindakan *drop* sehingga trafik tidak diteruskan ke DVWA dan mengha-

silkan *Alert* yang dicatat dalam *log*. *Log* tersebut diteruskan melalui *Wazuh Agent* ke *Wazuh Manager* untuk dianalisis dan dikorelasikan. Ketika kondisi *Active Response* terpenuhi, *Wazuh Manager* menjalankan *Firewall-drop* untuk memblokir alamat IP sumber melalui *Firewall*. Seluruh *Alert* dan aktivitas respons selanjutnya diindeks oleh *Wazuh Indexer* dan divisualisasikan melalui *Wazuh Dashboard* untuk pemantauan keamanan secara terpusat.

Alur proses deteksi dan mitigasi serangan pada sistem keamanan yang diusulkan ditunjukkan pada [Gambar 4](#). Trafik jaringan terlebih dahulu melewati *Firewall* dan diperiksa oleh *Suricata IPS* menggunakan aturan custom. Jika trafik tidak terindikasi sebagai serangan, *request* diteruskan ke *web server DVWA* untuk diproses dan menghasilkan *HTTP response* kepada pengguna.



Gambar 4. Flowchart Deteksi dan Mitigasi

Sebaliknya, apabila trafik terdeteksi mengandung pola *SQLi* atau *Stored XSS*, *Suricata* menghasilkan *Alert* yang dicatat dalam *log* dan diteruskan melalui *Wazuh Agent* ke *Wazuh Manager* untuk dianalisis serta dikorelasikan. Jika kondisi *Active Response* terpenuhi, *Wazuh* menjalankan *Firewall-drop* untuk memblokir alamat IP sumber melalui *Firewall*. Apabila kondisi tersebut tidak terpenuhi, *Alert* tetap dicatat tanpa pemblokiran otomatis, sedangkan hasil deteksi dan respons dipantau melalui *Wazuh Dashboard*.

Konfigurasi pengalamatan IP dilakukan secara statis untuk memastikan identifikasi sumber dan tujuan lalu lintas jaringan secara konsisten selama proses simulasi serangan dan analisis *log*. Konfigurasi tersebut menggunakan alamat IP kelas C dalam satu subnet, yaitu jaringan 192.168.100.0/24. Konfigurasi pengalamatan IP yang diterapkan dalam penelitian ini ditampilkan pada [Tabel 2](#).

Tabel 2. Pengalamatan IP pada Setiap VM

Nama VM	Alamat IP	Subnetmask	Fungsi
<i>Kali Attacker</i>	192.168.100.12	255.255.255.0	Mesin penyerang menjalankan SQLMap
<i>Cloud Server</i>	192.168.100.10	255.255.255.0	Server target menjalankan DVWA, <i>Suricata</i> , <i>Wazuh Agent</i> , dan <i>Firewall</i>
<i>Wazuh SIEM</i>	192.168.100.11	255.255.255.0	Server monitoring menjalankan <i>Wazuh Manager</i> , <i>Wazuh Indexer</i> , dan <i>Wazuh Dashboard</i>

2.2.3. Tahap Simulation Prototyping

Tahap *Simulation Prototyping* dilakukan dengan membangun lingkungan virtualisasi untuk mengevaluasi mekanisme deteksi dan mitigasi terhadap serangan pada DVWA. Pengujian mencakup tiga skenario *SQLi*, yaitu *Error-Based*, *Union-*

Based, dan Authentication Bypass, serta Stored XSS melalui penyisipan payload skrip menggunakan tools atau metode pengujian yang sesuai. Setiap skenario dilakukan dalam beberapa percobaan secara berulang. Setelah percobaan selesai dan alamat IP sumber terblokir, aturan *Firewall* dikosongkan menggunakan perintah *sudo iptables -F* sebelum percobaan berikutnya dilakukan. Pengulangan ini bertujuan mengamati konsistensi mekanisme deteksi dan mitigasi. Hasil pengujian menunjukkan bahwa setiap percobaan dapat mendeteksi dan menghentikan serangan serta memblokir alamat IP sumber, meskipun jumlah *Alert* dan entri pemblokiran pada *Firewall* dapat berbeda pada setiap percobaan.

2.2.4. Tahap Implementation

Tahap ini menerapkan dan mengonfigurasi seluruh arsitektur, meliputi Suricata dalam mode IPS, Wazuh *Manager*, Wazuh *Agent*, dan *Firewall* berbasis *iptables*. Suricata menggunakan aturan deteksi custom untuk SQLi dan *Stored XSS*, sedangkan Wazuh menerima *Alert log* dari Suricata, melakukan korelasi, dan menjalankan *Active Response* untuk menginstruksikan *Firewall* memblokir IP sumber secara otomatis.

3. Hasil

3.1. Hasil Sistem Keamanan Terintegrasi

Bagian ini menyajikan hasil implementasi dan verifikasi sistem keamanan yang terdiri atas *Cloud Server* DVWA, Suricata IPS, Wazuh SIEM, dan *Firewall*. Verifikasi dilakukan untuk memastikan setiap komponen telah terinstal, terkonfigurasi, dan beroperasi sebelum pengujian deteksi dan mitigasi dilakukan.

3.1.1. Hasil Implementasi Cloud Server

Cloud Server digunakan sebagai lingkungan untuk menjalankan DVWA dan layanan pendukungnya. Hasil implementasi *Ubuntu Server* dan DVWA ditunjukkan pada [Gambar 5](#).

```
mysql@cloud-server:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 24.04.4 LTS
Release:        24.04
Codename:       noble

mysql@cloud-server:~$ sudo systemctl status mysql
● mariadb.service - MariaDB 10.11.14 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; preset: enabled)
   Active: active (running) since Tue 2026-07-28 12:01:10 UTC; 2h 17min ago
     Docs: man:mariadb(8)
           https://mariadb.com/kb/en/library/systemd/
   Process: 8944 ExecStartPre=/usr/bin/install --yes -o mysql -g root -f /var/run/mysql (code=exited, status=0/SUCCESS)
   Process: 8781 ExecStartPost=/bin/bash -c 'systemctl unset-environment _WSREP_START_POSITION (code=exited, status=0/SUCCESS)'
   Process: 885 ExecStartPre=/bin/bash -c [ ! -e /usr/bin/galera_recovery ] && [[ $! = 0 ]] && /usr/bin/galera_recovery || { $? != 0 } && systemctl set-environment _WSREP_START_POSI...
   Process: 1038 ExecStartPost=/bin/bash -c 'systemctl unset-environment _WSREP_START_POSITION (code=exited, status=0/SUCCESS)'
   Process: 1039 ExecStartPost=/etc/mysql/debian-start (code=exited, status=0/SUCCESS)
 Main PID: 961 (mariadbd)
   Status: Taking your SQL requests now...
    Tasks: 0 (limit: 10937)
   Memory: 123.7M (peak: 127.1M)
      CPU: 7.46s
   CGroup: /system.slice/mariadb.service
           └─L661 /usr/sbin/mariadbd
```

```
Jul 28 12:01:23 cloud-server etc/mysql/debian-start[1636]
Jul 28 12:01:23 cloud-server etc/mysql/debian-start[1636]
Jul 28 12:01:23 cloud-server etc/mysql/debian-start[1636]
Jul 28 12:01:23 cloud-server etc/mysql/debian-start[1636]
Jul 28 12:01:23 cloud-server etc/mysql/debian-start[1636]
Jul 28 12:01:23 cloud-server etc/mysql/debian-start[1636]
Jul 28 12:01:23 cloud-server etc/mysql/debian-start[1636]
Jul 28 12:01:23 cloud-server etc/mysql/debian-start[1636]
Jul 28 12:01:23 cloud-server etc/mysql/debian-start[1636]
Jul 28 12:01:23 cloud-server etc/mysql/debian-start[1636]
Jul 28 12:01:23 cloud-server etc/mysql/debian-start[1636]
Jul 28 12:01:23 cloud-server etc/mysql/debian-start[1636]
Jul 28 12:01:23 cloud-server etc/mysql/debian-start[1636]
```

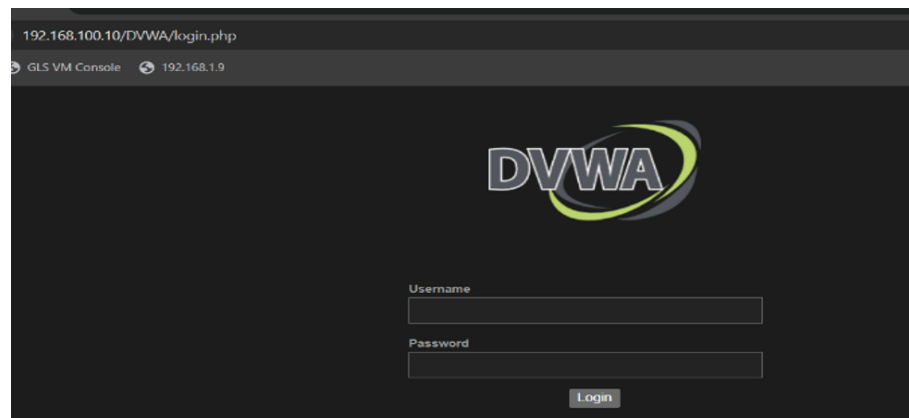
Improperly closed tables are also reported if clients are accessing the tables `#xxx`. A list of current connections is below.

	Id	User	Host	db	Command	Time	State	Info	Progress
	37	root	/localhost		Query	0	starting	show processlist	0.00%

Uptime: 15 s Threads: 1 Questions: 678 Slow queries: 0 Opens: 195 Open tables: 61 Queries per second avg: 45.200

Gambar 5. Implementasi *Ubuntu Server* dan *DVWA* pada *Cloud Server*

Cloud Server menggunakan *Ubuntu Server* 24.04.4 LTS dengan layanan Apache2 dan MariaDB dalam kondisi aktif. DVWA dapat diakses melalui *web browser* dan siap digunakan sebagai target pengujian. Halaman *login* DVWA ditunjukkan pada [Gambar 6](#), melalui alamat 192.168.100.10/DVWA.



Gambar 6. Halaman *login* DVWA

Halaman *login* dapat ditampilkan dengan baik, yang menunjukkan bahwa DVWA telah berjalan dan siap digunakan dalam pengujian.

3.1.2. Hasil Implementasi Wazuh SIEM

Wazuh diimplementasikan sebagai SIEM pada *Ubuntu Server 24.04.3 LTS* untuk mengumpulkan, menganalisis, mengorelasikan, dan merespons *log* keamanan secara terpusat. Status layanan Wazuh *Manager*, Wazuh *Indexer*, dan Wazuh *Dashboard* ditunjukkan pada Gambar 7.

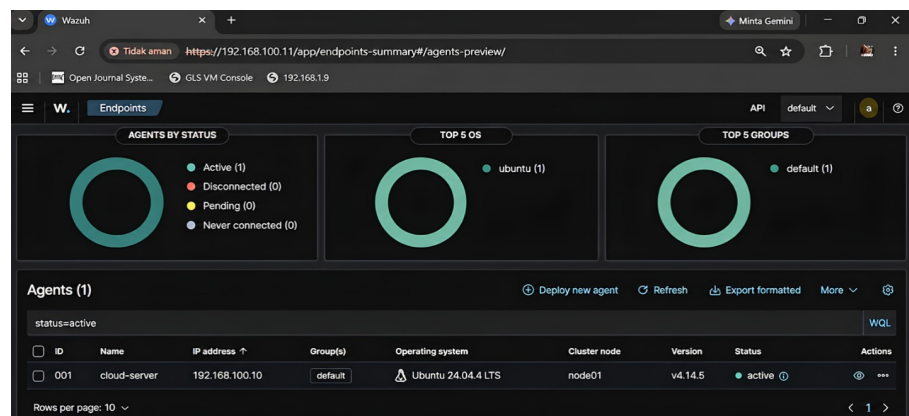
```

ardi2@wazuh-siem:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:   Ubuntu 24.04.3 LTS
Release:      24.04
Codename:     noble
ardi2@wazuh-siem:~$ systemctl is-active wazuh-manager wazuh-indexer wazuh-dashboard
active
active
active
ardi2@wazuh-siem:~$

```

Gambar 7. Status *Active* Wazuh *Manager*, Wazuh *Indexer*, dan Wazuh *Dashboard*

Ketiga layanan berada dalam kondisi *active (running)*. Kondisi tersebut menunjukkan bahwa Wazuh *Manager* siap menerima *log* dari *agent*, Wazuh *Indexer* menangani penyimpanan dan pengindeksan data keamanan, sedangkan Wazuh *Dashboard* menyediakan antarmuka untuk pemantauan dan visualisasi. Status endpoint yang terhubung dengan Wazuh *Manager* ditampilkan pada Gambar 8.



Gambar 8. Status Keaktifan Wazuh *Agent* pada *Interface* Wazuh *Manager*

Wazuh *Dashboard* menampilkan endpoint yang terintegrasi dengan Wazuh *Manager*. Informasi tersebut digunakan untuk memverifikasi konektivitas *agent* dan mendukung pemantauan keamanan secara terpusat.

3.1.3. Hasil Implementasi Suricata IPS

Suricata diimplementasikan sebagai IPS untuk melakukan inspeksi lalu lintas jaringan berdasarkan aturan yang telah dikonfigurasi. Status layanan Suricata ditunjukkan pada Gambar 9.

```

● suricata.service - Suricata IDS/IPS/NSM/FW daemon
   Loaded: loaded (/usr/lib/systemd/system/suricata.service; enabled; preset: enabled)
   Active: active (running) since Tue 2026-07-28 12:01:09 UTC; 2h 28min ago
     Docs: man:suricata(8)
           man:suricatasc(8)
           https://suricata.io/documentation/
   Process: 992 ExecStartPre=/bin/rm -f /run/suricata.pid (code=exited, status=0/SUCCESS)
   Main PID: 1018 (Suricata-Main)
    Tasks: 7 (Limit: 2263)
   Memory: 63.6M (peak: 63.9M)
      CPU: 2min 56.131s
   CGroup: /system.slice/suricata.service
           └─ 1018 /usr/bin/suricata --pid-file /run/suricata.pid --pid-file /run/suricata.pid

```

Gambar 9. Status *Active (running)* Layanan Suricata

Layanan Suricata berada dalam kondisi *active (running)*, sehingga sistem siap melakukan inspeksi lalu lintas dan menerapkan aturan deteksi yang telah dikonfigurasi.

Konfigurasi aturan *custom rule* terlihat pada Gambar 10 yang digunakan untuk mendeteksi dan menangani SQLi dan *Stored XSS*.

```

root@cloud-server:~# cat /etc/suricata/rules/local.rules
# rules SQL Injection
drop http any any -> 192.168.100.10 $HTTP_PORTS (msg:"ET WEB_SERVER Possible SQL Injection Attempt UNION SELECT"; flow:established,to_server; content:"UNION"; http_uri; nocase; content:"SELECT"; http_uri; nocase; pcre:"/UNION.*SELECT/ui"; reference:url:doc.emergingthreats.net/2006446; classtype:web-application-attack; sid:2006447; rev:13; metadata:affected_product Web_Server_Applications, attack_target Web_Server, created_at 2010_07_30, deployment Datacenter, signature_severity Major; tag SQL_Injection, updated_at 2020_09_01;)
drop tcp any any -> 192.168.100.10 $HTTP_PORTS (msg:"SQL Injection Detected"; flow:established,to_server; content:"id"; nocase; http_uri; pcre:"/(and|wselect|union.*select)|or|and|and.+0x(''|\"'')/ui"; classtype:web-application-attack; sid:1000005; rev:1;)
drop tcp any any -> 192.168.100.10 $HTTP_PORTS (msg:"SQL Injection - Start Attacks - - - SQL"; /cre:"/(((((\%6f)|o)|(\%4f)|(\%52)|c2)|(\%72)|(\%20))|((\%67)|o)|(\%49)|((\%53)|s)|(\%73)|(\%20)|((\%6f)|t)|(\%4c)|((\%65)|\%4e)|e|(\%45)|\%3)|\%b|(\%20)|(\%74)|(\%74)|t|(\%54)|t|(\%73)|f|(\%30)|s|(\%53)|(\%69)|q|(\%49)|(\%6f)|o|(\%4f)|(\%6e)|n|(\%4e)|(\%73)|s|(\%53)|(\%71)|q|(\%51)|(\%6c)|l|(\%4c)|(\%4c)|t|(\%20)|(\%6e)|i|(\%49)|(\%4e)|n|(\%4e)|6a)|s)|(\%4a)|(\%65)|o|(\%45)|(\%63)|c|(\%43)|c|(\%74)|t|(\%54)|i|(\%69)|(\%49)|(\%6f)|o|(\%4f)|(\%4f)|(\%4e)|n|(\%4e)|((\%20)|f|(\%20)|-)|(\%27)|'(\%48)|h|(\%48)|(\%61)|a|(\%41)|(\%76)|v|(\%76)|(\%69)|i|(\%49)|(\%6e)|n|(\%4e)|(\%67)|g|(\%47)|(\%40)|(\%20)|((\%31)|1)|(\%3d)|+)|(\%31)|12))|(\%31)|(\%53)|5)|(\%3c)|<|(\%27)|'|1'|(\%7c)|(\%66)|f|(\%6f)|o|(\%67)|i"; classtype:web-application-attack; sid:1; rev:1;)

#rule stored XSS
drop http any any -> 192.168.100.10 $HTTP_PORTS (msg:"Stored XSS Detected - URL Encoded Script Tag"; flow:established,to_server; content:"/DWA/vulnerabilities/xss.s/"; http_uri; content:"POST"; http_method; file_data; content:"script"; nocase; pcre:"/((\%3C)script/i"; classtype:web-application-attack; sid:1000010; rev:3;)
drop http any any -> 192.168.100.10 $HTTP_PORTS (msg:"Stored XSS Detected - URL Encoded Script Tag"; flow:established,to_server; content:"/DWA/vulnerabilities/xss.s/"; http_uri; content:"POST"; http_method; content:"script"; nocase; http_client_body; pcre:"/(<|\%3C)script/i"; classtype:web-application-attack; sid:1000011; rev:3;)

```

Gambar 10. *Custom rule* SQLi dan *Stored XSS*

Konfigurasi deteksi SQLi terdiri atas tiga komponen. Pertama, *signature* ET WEB_SERVER Possible SQLi Attack UNION SELECT dengan SID 2006447 untuk mendeteksi pola UNION SELECT. Kedua, aturan SQLi Start Attacks 1....-SQLi dengan SID 1000005 yang menggunakan http_uri dan PCRE untuk mendeteksi variasi pola SQLi. Ketiga, tindakan *drop* diterapkan terhadap lalu lintas yang memenuhi kondisi deteksi. Ketiga komponen tersebut membentuk mekanisme deteksi dan pencegahan SQLi melalui pencocokan *signature* dan regular expression, yang dilanjutkan dengan tindakan *drop* terhadap lalu lintas terindikasi serangan.

Deteksi *Stored XSS* menggunakan SID 1000010 untuk mendeteksi *script tag*, termasuk pola URL *encoded*, dan SID 1000011 untuk memeriksa *request body* menggunakan pola *script* dan PCRE guna mengidentifikasi payload JavaScript. Kedua aturan menerapkan tindakan *drop* terhadap lalu lintas yang memenuhi karakteristik *Stored XSS*.

3.1.4. Hasil Implementasi Active Response

Integrasi Suricata dan Wazuh diverifikasi melalui konfigurasi pembacaan *log* dan *Active Response*. Konfigurasi pembacaan *log* Suricata oleh Wazuh Agent ditunjukkan pada Gambar 11.

```

GNU nano 7.2

<localfile>
  <log_format>syslog</log_format>
  <location>/var/log/suricata/fast.log</location>
</localfile>

<localfile>
  <log_format>json</log_format>
  <location>/var/log/suricata/eve.json</location>
</localfile>

```

Gambar 11. Konfigurasi Pembacaan *log* Suricata oleh Wazuh Agent

Berkas `/var/log/Suricata/fast.log` dikonfigurasi sebagai sumber *log* berformat *syslog*, sedangkan `/var/log/Suricata/eve.json` sebagai sumber *log* berformat JSON. Kedua sumber tersebut dibaca oleh *Wazuh Agent* dan diteruskan ke *Wazuh Manager* untuk mendukung pemantauan, analisis, dan korelasi *event* keamanan yang dihasilkan Suricata. Selanjutnya konfigurasi *rule* Wazuh untuk SQLi dan *Stored XSS* ditunjukkan pada [Gambar 12](#).

```

GNU nano 7.2 /var/ossec/etc/rules/local_rules.xml *
<group name="penelitian_ardi">
  <!-- Rule Spesifik SQLi serangan Bertubi -->
  <rule id="100005" level="12">
    <if_matched_sid>31106</if_matched_sid>
    <match>select|union|order by|where|extractvalue|updatexml|floor(rand|cast|convert|(|OR 1=1|OR '1'='1|vulnerabilities/sqli|login|.php|sqlmap|</match>
    <same_source_ip />
    <description>Active Response Terdeteksi Serangan Berkelanjutan SQL Injection Web Server</description>
    <group>web_attack,sqli_injection</group>
  </rule>
</group>

  <!-- Rule Spesifik XSS -->
  <rule id="100060" level="12">
    <if_matched_sid>31106</if_matched_sid>
    <match>script|alert|document\\.cookie|document\\.body\\.innerHTML|xss_s|sqlmap|onerror|javascript|xsstrike|\\3Cscript%3E|onload=</match>
    <same_source_ip />
    <description>Active Response Terdeteksi Serangan XSS Berulang pada Web Server</description>
    <group>web_attack,xss</group>
  </rule>
</group>

```

Gambar 12. Konfigurasi *Rule* Wazuh Manager untuk Deteksi SQLi dan *Stored XSS*

Rule pada [Gambar 12](#) berfungsi sebagai mekanisme deteksi lanjutan untuk mengidentifikasi SQLi dan *Stored XSS* berulang dari sumber yang sama. *Rule* ID 100005 (level 12) menggunakan `<if_matched_sid>31106</if_matched_sid>`, `<match>` untuk mencocokkan pola serangan, dan `<same_source_ip />` untuk memastikan kesamaan alamat IP sumber. Sementara itu, *Rule* ID 100060 (level 12) menggunakan mekanisme serupa dengan pola yang diarahkan pada karakteristik *Stored XSS*, seperti *script*, *Alert*, *java-script*, dan `%3Cscript%3E`. Kedua *rule* berfungsi menghasilkan *Alert* dan klasifikasi serangan, bukan melakukan pemblokiran secara langsung. Pemblokiran alamat IP dilakukan melalui mekanisme Wazuh *Active Response* yang dikonfigurasi secara terpisah. Dengan demikian, *rule* tersebut memperkuat deteksi serangan berdasarkan *event* pemicu, pola serangan, dan kesesuaian IP sumber.

Konfigurasi Wazuh *Active Response* yang menghubungkan *rule* deteksi dengan tindakan pemblokiran *Firewall-drop* ditampilkan pada [Gambar 13](#). Konfigurasi ini merupakan bagian dari mekanisme mitigasi, di mana Wazuh Manager menjalankan *Firewall-drop* terhadap alamat IP sumber ketika *event* memenuhi kondisi *Rule* ID 100005 untuk SQLi atau *Rule* ID 100060 untuk *Stored XSS*.

```

GNU nano 7.2

<active-response>
  <command>firewall-drop</command>
  <location>local</location>
  <rules_id>100005,100060</rules_id>
  <timeout>0</timeout>
</active-response>

```

Gambar 13. Konfigurasi *Firewall-drop Active Response*

Dengan demikian, *Active Response* berperan meneruskan hasil deteksi Wazuh menjadi tindakan pemblokiran pada *Firewall*. **Gambar 13** menunjukkan mekanisme tersebut, di mana `<command>Firewall-drop</command>` menentukan perintah pemblokiran alamat IP sumber, sedangkan `<location>local</location>` menunjukkan bahwa tindakan dijalankan pada mesin lokal. Konfigurasi `<rule_id>100005,100060</rule_id>` menghubungkan *Active Response* dengan *Rule ID* 100005 untuk *SQLi* dan *Rule ID* 100060 untuk *Stored XSS*. Kedua *rule* tersebut merupakan aturan korelasi dan deteksi Wazuh yang berfungsi sebagai pemicu *Active Response*, bukan *rule* Suricata. Pada implementasinya, mitigasi dipicu secara langsung oleh terpenuhinya kondisi pada aturan Wazuh saat memproses data *log* Apache yang dikirimkan oleh *wazuh agent*. Hasilnya ketika kondisi *rule* terpenuhi, alamat IP sumber diteruskan ke *Firewall-drop* untuk diblokir. Sementara itu, `<timeout>0</timeout>` menunjukkan bahwa tindakan pemblokiran tidak memiliki batas waktu otomatis berdasarkan parameter *timeout*.

3.2. Hasil Pengujian Sistem Setelah Penerapan Keamanan

Pengujian dilakukan terhadap empat skenario, yaitu *Error-Based SQLi*, *Union-Based SQLi*, *Authentication Bypass SQLi*, dan *Stored XSS*. Pengujian bertujuan mengevaluasi dan mencakup kemampuan Suricata dalam mendeteksi dan menajutuhkan trafik, Wazuh dalam mencatat *Alert*, *Active Response* dalam menjalankan respons otomatis, serta *Firewall* dalam memblokir alamat IP sumber. Verifikasi pada sisi *server* dilakukan menggunakan *fast.log* karena formatnya lebih sederhana dan mudah dibaca secara manual dalam bentuk teks, sehingga mendukung proses audit atau inspeksi *log* secara cepat oleh peneliti sebagaimana dikonfigurasi pada **Gambar 11**.

3.2.1. Hasil Pengujian *SQLi*

Pengujian *Error-Based SQLi* dilakukan dari Kali Linux 192.168.100.12 terhadap DVWA pada 192.168.100.10. Nmap mengonfirmasi port 80/tcp terbuka, kemudian SQLMap menguji parameter *id* menggunakan teknik *Error-Based injection*. Aktivitas pengujian ditunjukkan pada **Gambar 14**.

```

[card13@ard1 ~]$ nmap -p 80 192.168.100.10
Nmap scan report for 192.168.100.10 (192.168.100.10)
Host is up (0.0025s latency).

PORT      STATE SERVICE
80/tcp    open  http
MAC Address: 08:00:27:84:F1:F6 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.53 seconds

[card13@ard1 ~]$ sqlmap -u "http://192.168.100.10/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie "PHPSESSID=rvvdqnfadt1d5orondr4vt14qpu; security
ables=1 users --columns --batch

[+] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to o
s. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 03:30:20 /2026-09-10/

[03:30:20] [INFO] flushing session file
[03:30:20] [INFO] testing connection to the target URL
[03:30:21] [INFO] checking if the target is protected by some kind of WAF/IPS
[03:30:21] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQL')
[03:30:21] [INFO] testing for SQL injection on GET parameter 'id'
[03:30:21] [INFO] it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [y/n] Y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [y/n] Y
[03:30:21] [WARNING] reflective value(s) found and filtering out
[03:30:22] [WARNING] there is a possibility that the target (or WAF/IPS) is dropping 'suspicious' requests
[03:30:22] [CRITICAL] connection timed out to the target URL. sqlmap is going to retry the request(s)
[03:32:22] [CRITICAL] connection timed out to the target URL. sqlmap is going to retry the request(s)
[03:32:22] [CRITICAL] connection timed out to the target URL. sqlmap is going to retry the request(s)
[03:34:23] [CRITICAL] connection timed out to the target URL. sqlmap is going to retry the request(s)
[03:34:23] [CRITICAL] connection timed out to the target URL. sqlmap is going to retry the request(s)
[03:36:23] [CRITICAL] connection timed out to the target URL. sqlmap is going to retry the request(s)
[03:36:23] [CRITICAL] connection timed out to the target URL. sqlmap is going to retry the request(s)

```

Gambar 14. Pengujian *Error-Based SQLi* Menggunakan SQLMap

Pengujian dimulai pada 03:30:20, ketika parameter id teridentifikasi sebagai kandidat yang berpotensi rentan terhadap injeksi SQLi. Pada 03:30:52, SQL-Map menampilkan peringatan mengenai kemungkinan mekanisme perlindungan WAF/IPS pada sistem target. Selanjutnya, pada 03:30:52–03:36:53, muncul pesan “connection timed out to the target URL” secara berulang, sehingga SQLMap tidak dapat melanjutkan pengujian secara optimal.

Pencatatan *Alert* Suricata dan aturan *iptables* diperiksa untuk memverifikasi deteksi serta pemblokiran alamat IP sumber, sebagaimana ditunjukkan pada Gambar 15.

```
ardil@cloud-server:~$ sudo iptables -F
ardil@cloud-server:~$ sudo iptables -L INPUT -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source                   destination
ardil@cloud-server:~$ sudo cp /dev/null /var/log/suricata/fast.log
ardil@cloud-server:~$ sudo tail -f /var/log/suricata/fast.log
09/10/2026-03:30:23.586631 [wDrop] [**] [1:1000005:1] SQL Injection Detected [**] [Classification: Web Application Attack] [Priority: 1] [TCP] 192.168.100.12:48186 -> 192.168.100.10:80
09/10/2026-03:30:23.586631 [wDrop] [**] [1:2006447:13] ET_WEB_SERVER Possible SQL Injection Attempt UNION SELECT [**] [Classification: Web Application Attack] [Priority: 1] [TCP] 192.168.100.12:48186 -> 192.168.100.10:80
09/10/2026-03:30:23.983962 [wDrop] [**] [1:1000005:1] SQL Injection Detected [**] [Classification: Web Application Attack] [Priority: 1] [TCP] 192.168.100.12:48234 -> 192.168.100.10:80
09/10/2026-03:30:23.983962 [wDrop] [**] [1:1000005:1] SQL Injection Detected [**] [Classification: Web Application Attack] [Priority: 1] [TCP] 192.168.100.12:48259 -> 192.168.100.10:80
09/10/2026-03:30:24.085117 [wDrop] [**] [1:1000005:1] SQL Injection Detected [**] [Classification: Web Application Attack] [Priority: 1] [TCP] 192.168.100.12:48266 -> 192.168.100.10:80
09/10/2026-03:30:24.169451 [wDrop] [**] [1:1000005:1] SQL Injection Detected [**] [Classification: Web Application Attack] [Priority: 1] [TCP] 192.168.100.12:48270 -> 192.168.100.10:80
09/10/2026-03:30:24.291727 [wDrop] [**] [1:1000005:1] SQL Injection Detected [**] [Classification: Web Application Attack] [Priority: 1] [TCP] 192.168.100.12:48278 -> 192.168.100.10:80
^C
ardil@cloud-server:~$ sudo iptables -L INPUT -n -v
Chain INPUT (policy ACCEPT 461 packets, 32680 bytes)
 pkts bytes target     prot opt in     out     source                   destination
281 19415 DROP      0    --  *      *      192.168.100.12          0.0.0.0/0
5 2376 DROP      0    --  *      *      192.168.100.12          0.0.0.0/0
0 0 DROP      0    --  *      *      192.168.100.12          0.0.0.0/0
0 0 DROP      0    --  *      *      192.168.100.12          0.0.0.0/0
2 639 DROP      0    --  *      *      192.168.100.12          0.0.0.0/0
ardil@cloud-server:~$
```

Gambar 15. Hasil Deteksi Suricata dan Pemblokiran oleh *Firewall*

Berdasarkan pencatatan Suricata, mencatat 7 kejadian deteksi pada 03:30:23.586631–03:30:24.291727. Enam kejadian menggunakan SID 1000005 dengan deskripsi “SQLi Detected”, sedangkan satu kejadian menggunakan SID 2006447 dengan deskripsi “ET WEB_SERVER Possible SQLi Attempt UNION SELECT”. Seluruh kejadian berstatus [wDrop] dari 192.168.100.12 menuju 192.168.100.10:80. Pemeriksaan *iptables* menunjukkan aturan DROP terhadap IP sumber dengan *counter* tertinggi 281 paket dan 19.415 bytes, yang menunjukkan bahwa lalu lintas pengujian dijatuhkan oleh Suricata dan alamat IP sumber diblokir pada *Firewall*.

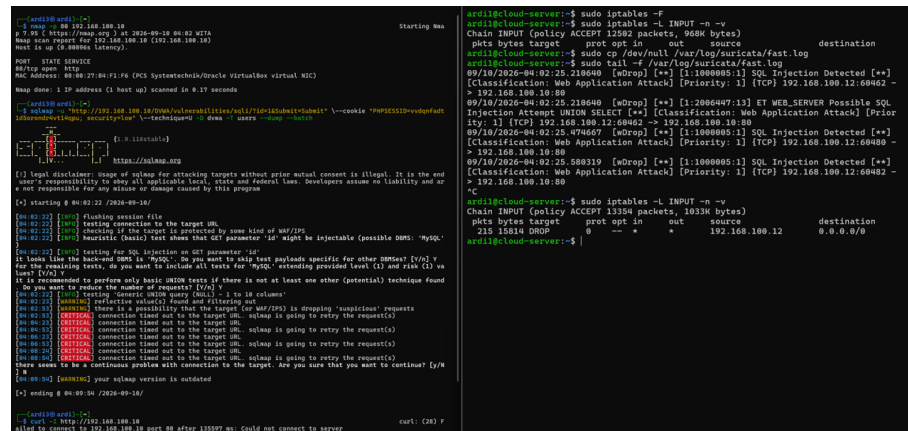
Hasil pemantauan Wazuh *Dashboard* untuk mengamati urutan *Alert* Suricata, *Active Response*, dan *Firewall-drop*, sebagaimana ditunjukkan pada Gambar 16.

W. Threat Hunting cloud-server					
13,322 hits					
Sep 9, 2026 @ 03:36:31.093 - Sep 10, 2026 @ 03:36:31.093					
Export Formatted Reset view 786 available fields Columns Density 1 fields sorted Exit full screen					
timestamp	agent.name	rule.description	rule.level	rule.id	
Sep 10, 2026 @ 03:30:28.223	cloud-server	Host Blocked by firewall-drop Active Response	3	651	
Sep 10, 2026 @ 03:30:26.302	cloud-server	Host Blocked by firewall-drop Active Response	3	651	
Sep 10, 2026 @ 03:30:26.276	cloud-server	Host Blocked by firewall-drop Active Response	3	651	
Sep 10, 2026 @ 03:30:26.259	cloud-server	Host Blocked by firewall-drop Active Response	3	651	
Sep 10, 2026 @ 03:30:26.242	cloud-server	Host Blocked by firewall-drop Active Response	3	651	
Sep 10, 2026 @ 03:30:26.229	cloud-server	Active Response Terdeteksi Serangan Berkelanjutan SQL Injection Web ...	12	100005	
Sep 10, 2026 @ 03:30:26.218	cloud-server	Suricata: Alert - SQL Injection Detected	3	86601	
Sep 10, 2026 @ 03:30:24.291	cloud-server	Active Response Terdeteksi Serangan Berkelanjutan SQL Injection Web ...	12	100005	
Sep 10, 2026 @ 03:30:24.290	cloud-server	Active Response Terdeteksi Serangan Berkelanjutan SQL Injection Web ...	12	100005	
Sep 10, 2026 @ 03:30:24.286	cloud-server	Active Response Terdeteksi Serangan Berkelanjutan SQL Injection Web ...	12	100005	
Sep 10, 2026 @ 03:30:24.282	cloud-server	Active Response Terdeteksi Serangan Berkelanjutan SQL Injection Web ...	12	100005	
Sep 10, 2026 @ 03:30:24.274	cloud-server	A web attack returned code 200 (success).	6	31106	
Sep 10, 2026 @ 03:30:24.263	cloud-server	Suricata: Alert - SQL Injection Detected	3	86601	
Sep 10, 2026 @ 03:30:24.258	cloud-server	Suricata: Alert - SQL Injection Detected	3	86601	
Sep 10, 2026 @ 03:30:24.249	cloud-server	Suricata: Alert - SQL Injection Detected	3	86601	

Gambar 16. *Event* Wazuh *Dashboard* pada Pengujian *Error-Based* SQLi

Wazuh mencatat *event* pada 03:30:24.249–03:30:28.223, terdiri atas 4 *Alert* Suricata (*Rule* ID 86601), 5 *Active Response* (*Rule* ID 100005, level 12), 5 *Firewall-drop* (*Rule* ID 651), dan 1 *event* HTTP 200 (*Rule* ID 31106, level 6). Berdasarkan timestamp, *Alert Suricata* pertama tercatat pada 03:30:24.249, diikuti *Active Response* pada 03:30:24.282–03:30:24.291 dan *Firewall-drop* pada 03:30:26.242–03:30:28.223. Urutan tersebut menunjukkan bahwa deteksi Suricata diteruskan ke Wazuh dan memenuhi kondisi *rule* untuk memicu *Active Response*, yang selanjutnya diikuti penerapan *Firewall-drop* terhadap alamat IP sumber. Dengan demikian, tahapan deteksi, respons otomatis, dan pemblokiran dapat diamati secara berurutan melalui pencatatan *event*.

Pengujian *Union-Based* SQLi dilakukan menggunakan SQLMap terhadap parameter id pada DVWA dengan opsi `-technique=U`. Aktivitas pengujian ditunjukkan pada [Gambar 17](#).



Gambar 17. Pengujian *Union-Based* SQLi Menggunakan SQLMap

SQLMap mengidentifikasi parameter id sebagai parameter yang dapat diinjeksi pada DBMS MySQL. Pada pengujian Generic UNION query, SQLMap mendeteksi indikasi mekanisme pemblokiran WAF/IPS. Suricata kemudian mencatat 4 *Alert* pada 04:02:25.210640–04:02:25.580319, terdiri atas 3 kejadian SID 1000005 dan 1 kejadian SID 2006447, seluruhnya berstatus [wDrop], sumber 192.168.100.12 menuju 192.168.100.10 pada port 80. Setelah pemblokiran, SQLMap mengalami connection timeout, sedangkan cURL juga tidak dapat terhubung ke port 80. Aturan *iptables* menunjukkan DROP terhadap 192.168.100.12 dengan *counter* 215 paket dan 15.814 bytes.

Pencatatan *log* dan mekanisme respons terhadap *Union-Based* SQLi dianalisis melalui Wazuh *Dashboard* pada [Gambar 18](#) dengan berfokus pada *Alert* Suricata, *Active Response*, dan *Firewall-drop*.

terhadap alamat IP sumber masih aktif setelah pengujian selesai, sebagaimana ditunjukkan pada [Gambar 20](#).

[illegible]

Gambar 20. Hasil Deteksi Suricata dan Pemeriksaan Aturan *iptables*

Suricata mencatat 46 kejadian berstatus [wDrop], terdiri atas 1 *event* “ET WEB_SERVER Possible SQLi Attack UNION SELECT” dan 45 *event* “SQLi - Start Attacks 1....-SQLi”, seluruhnya berasal dari 192.168.100.12 menuju 192.168.100.10:80. Pemeriksaan *iptables* setelah pengujian tidak menunjukkan aturan DROP terhadap alamat IP 192.168.100.12. Temuan ini menunjukkan bahwa lalu lintas pengujian terdeteksi dan dijatuhkan oleh Suricata, tetapi pada saat pemeriksaan dilakukan tidak ditemukan bukti pemblokiran alamat IP sumber melalui *Firewall*.

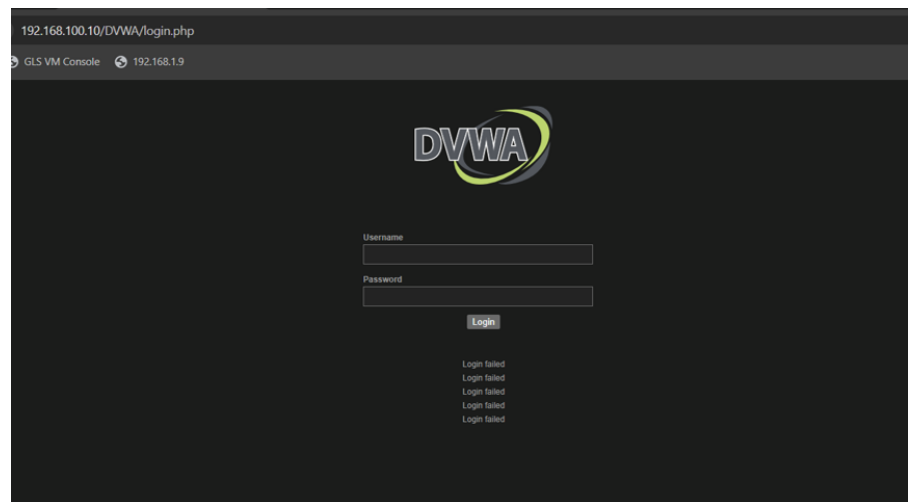
Hasil deteksi Suricata selanjutnya dianalisis melalui Wazuh *Dashboard* untuk mengamati pemrosesan *log* menjadi *Alert* pada sistem pemantauan. Analisis difokuskan pada jumlah dan waktu *event* yang diterima oleh agen *cloud-server*. Hasil pemantauan tersebut ditampilkan pada [Gambar 21](#).

W. Threat Hunting		cloud-server	API	default	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	
		cloud-server		API	default	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
		cloud-server		API	default	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
		cloud-server		API	default	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
		cloud-server		API	default	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
		cloud-server		API	default	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
		cloud-server		API	default	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
		cloud-server		API	default	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
		cloud-server		API	default	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
		cloud-server		API	default	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
		cloud-server		API	default	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
		cloud-server		API	default	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
		cloud-server		API	default	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
		cloud-server		API	default	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
		cloud-server		API	default	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
		cloud-server		API	default	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
		cloud-server		API	default	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100

Gambar 21. *Alert Suricata pada Wazuh Dashboard*

Wazuh mencatat 15 *Alert* dengan deskripsi “Suricata: Alert – SQLi - Start Attacks 1....- SQLi”, level 3, dan *Rule* ID 86601 pada rentang 00:26:50.558–00:26:53.776. Perbedaan jumlah dengan 46 *event* pada *log* Suricata berkaitan dengan cakupan data yang ditampilkan, karena *Dashboard* hanya menampilkan *event* dalam rentang waktu dan tampilan yang diamati. Dengan demikian, *log* Suricata merepresentasikan keseluruhan kejadian yang tercatat, sedangkan Wazuh *Dashboard* menampilkan *event* yang telah diproses menjadi *Alert*.

Verifikasi terhadap halaman autentikasi DVWA dilakukan setelah pengujian, sebagaimana ditunjukkan pada [Gambar 22](#).

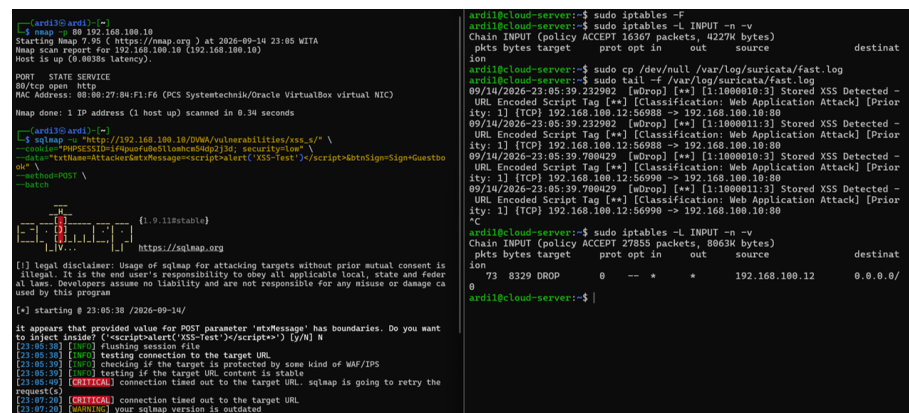


Gambar 22. Halaman *login* DVWA Setelah Pengujian *Authentication Bypass*

Halaman *login* DVWA masih menampilkan pesan “*Login failed*” sebanyak lima kali. Tidak ditemukan indikator keberhasilan *login*, pengalihan halaman, maupun akses melalui mekanisme *bypass*. Kondisi tersebut konsisten dengan hasil SQLMap yang tidak mengindikasikan injeksi pada parameter *username*.

3.2.2. Hasil Pengujian *Stored XSS*

Pengujian *Stored XSS* dilakukan dari *Kali Attacker* 192.168.100.12 terhadap DVWA pada 192.168.100.10:80. Nmap memverifikasi ketersediaan layanan HTTP pada port 80. Selanjutnya, payload `<script>Alert('XSS-Test')</script>` dikirimkan melalui parameter *mtxMessage* menggunakan metode POST. Aktivitas pengujian ditunjukkan pada Gambar 23.



Gambar 23. Pengujian *Stored XSS* Menggunakan SQLMap

SQLMap mengalami connection timeout setelah pengiriman payload. Suricata mencatat 4 *Alert* pada 23:05:39.232902–23:05:39.700429, terdiri atas 2 *Alert* SID 1000010 dan 2 *Alert* SID 1000011, seluruh *Alert* berstatus [wDrop] dari 192.168.100.12 menuju 192.168.100.10:80. Pemeriksaan *iptables* menunjukkan bahwa sebelum pengujian belum terdapat aturan DROP, sedangkan setelah pengujian IP 192.168.100.12 diblokir dengan *counter* 73 paket dan 8.329 bytes. Data tersebut menunjukkan bahwa lalu lintas pengujian terdeteksi dan dijatuhkan oleh Suricata, kemudian alamat IP sumber diblokir melalui mekanisme *Active Response* pada *Firewall*.

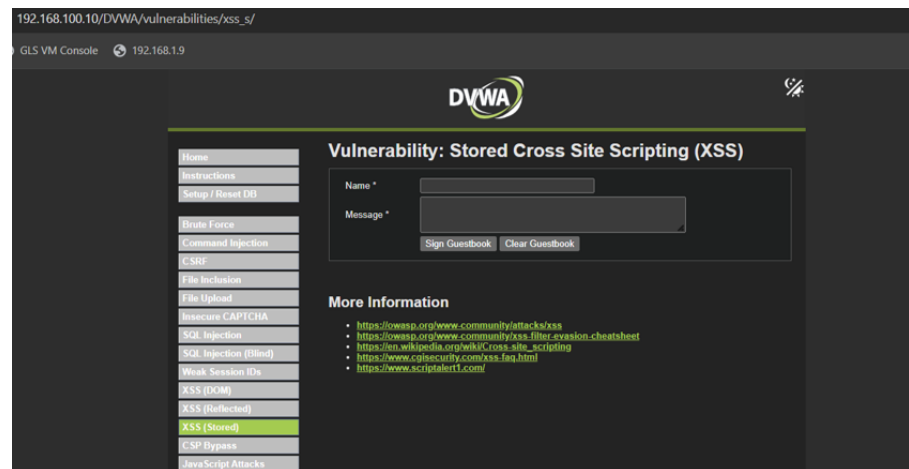
Respons sistem terhadap *request* selama pengujian dianalisis melalui Wazuh *Dashboard* berdasarkan *event* keamanan yang tercatat, sebagaimana ditunjukkan

pada Gambar 24.

timestamp	agent.name	rule.description	rule.level	rule.id
Sep 14, 2026 @ 23:05:42.127	cloud-server	Host Blocked by firewall-drop Active Response	3	651
Sep 14, 2026 @ 23:05:42.108	cloud-server	Host Blocked by firewall-drop Active Response	3	651
Sep 14, 2026 @ 23:05:42.102	cloud-server	Host Blocked by firewall-drop Active Response	3	651
Sep 14, 2026 @ 23:05:42.086	cloud-server	Host Blocked by firewall-drop Active Response	3	651
Sep 14, 2026 @ 23:05:42.079	cloud-server	Host Blocked by firewall-drop Active Response	3	651
Sep 14, 2026 @ 23:05:40.097	cloud-server	Active Response Terdeteksi Serangan XSS Berulang pada Web Server	12	100060
Sep 14, 2026 @ 23:05:40.086	cloud-server	Suricata: Alert - Stored XSS Detected - URL Encoded Script Tag	3	86601
Sep 14, 2026 @ 23:05:40.084	cloud-server	Suricata: Alert - Stored XSS Detected - URL Encoded Script Tag	3	86601
Sep 14, 2026 @ 23:05:40.073	cloud-server	Suricata: Alert - Stored XSS Detected - URL Encoded Script Tag	3	86601
Sep 14, 2026 @ 23:05:40.070	cloud-server	Suricata: Alert - Stored XSS Detected - URL Encoded Script Tag	3	86601
Sep 14, 2026 @ 23:05:36.112	cloud-server	PAM: Login session opened.	3	5501
Sep 14, 2026 @ 23:05:36.065	cloud-server	Successful sudo to ROOT executed.	3	5402
Sep 14, 2026 @ 23:05:24.063	cloud-server	PAM: Login session closed.	3	5502
Sep 14, 2026 @ 23:05:24.063	cloud-server	PAM: Login session opened.	3	5501
Sep 14, 2026 @ 23:05:24.060	cloud-server	Successful sudo to ROOT executed.	3	5402

Gambar 24. Event Wazuh Dashboard pada Pengujian Stored XSS

Wazuh mencatat 4 *Alert Suricata* (Rule ID 86601) pada 23:05:40.070–23:05:40.086. Kemudian *Active Response* dengan Rule ID 100060, level 12 terpicu pada 23:05:40.097, sekitar 27 ms setelah *Alert* terakhir. Selanjutnya, *Firewall-drop* dengan Rule ID 651 tercatat pada 23:05:42.079–23:05:42.127, dengan jeda sekitar 1,982 detik setelah *Active Response*. Urutan tersebut menunjukkan proses deteksi, pemicu *Active Response*, dan penerapan pemblokiran melalui *Firewall-drop*. Setelah pengujian, halaman Vulnerability: *Stored XSS* pada DVWA diperiksa kembali, sebagaimana ditunjukkan pada Gambar 25.



Gambar 25. Kondisi halaman DVWA setelah pengujian Stored XSS

Halaman Vulnerability: *Stored XSS* masih dapat diakses setelah pengujian, dengan kolom Name dan Message dalam kondisi kosong serta tidak menampilkan payload maupun eksekusi JavaScript. Kondisi tersebut tidak menunjukkan bukti visual keberhasilan eksekusi *Stored XSS*. Temuan ini konsisten dengan pencatatan Wazuh yang menunjukkan adanya deteksi Suricata, pemicu *Active Response*, dan pemblokiran alamat IP sumber.

3.2.3. Rekapitulasi Hasil Pengujian

Untuk mengevaluasi kinerja sistem keamanan terintegrasi, dilakukan pengujian terhadap empat skenario serangan, yaitu *Error-Based SQLi*, *Union-Based SQLi*, *Authentication Bypass SQLi*, dan *Stored XSS*. Evaluasi mencakup kemam-

puan Suricata IPS dalam mendeteksi dan menjatuhkan trafik, Wazuh SIEM dalam mencatat *Alert*, *Active Response* dalam menjalankan respons otomatis, serta *Firewall* dalam memblokir alamat IP sumber. Rekapitulasi hasil pengujian ditampilkan pada Tabel 3.

Tabel 3. Rekapitulasi Hasil Pengujian Sistem Keamanan

Skenario Pengujian	Suricata IPS	Wazuh SI-EM	Active Response	Res-	Firewall
<i>Error-Based SQLi</i>	Terdeteksi (<i>Drop</i>)	<i>Alert</i> berhasil dicatat	Berhasil dija-lankan	dija-	IP diblokir
<i>Union-Based SQLi</i>	Terdeteksi (<i>Drop</i>)	<i>Alert</i> berhasil dicatat	Berhasil dija-lankan	dija-	IP diblokir
<i>Authentication Bypass SQLi</i>	Terdeteksi (<i>Drop</i>)	<i>Alert</i> berhasil dicatat	Tidak dija-lankan	dija-	Tidak
<i>Stored XSS</i>	Terdeteksi (<i>Drop</i>)	<i>Alert</i> berhasil dicatat	Berhasil dija-lankan	dija-	IP diblokir

Berdasarkan Tabel 3, Suricata mendeteksi dan menjatuhkan trafik pada seluruh skenario pengujian, sedangkan Wazuh mencatat *Alert* dari aktivitas yang terdeteksi. Pada *Error-Based SQLi*, *Union-Based SQLi*, dan *Stored XSS*, kondisi *rule* terpenuhi sehingga *Active Response* dijalankan dan diikuti pemblokiran alamat IP sumber melalui *Firewall*. Sementara itu, pada *Authentication Bypass SQLi*, *Active Response* tidak dijalankan dan tidak terdapat aturan DROP pada *Firewall*. Kondisi tersebut sejalan dengan hasil SQLMap yang tidak mengidentifikasi parameter username sebagai parameter yang dapat diinjeksi dan tidak menunjukkan keberhasilan *bypass* autentikasi. Hasil tersebut menunjukkan bahwa tindakan mitigasi otomatis bergantung pada terpenuhinya kondisi *rule* yang telah dikonfigurasi.

4. Pembahasan

Untuk memberikan gambaran mengenai kinerja setiap komponen pada sistem keamanan terintegrasi, dilakukan analisis terhadap hasil implementasi berdasarkan fungsi yang dijalankan. Analisis ini mencakup kemampuan masing-masing komponen dalam mendukung proses deteksi, analisis, pemantauan, dan mitigasi serangan. Ringkasan hasil implementasi dan analisis setiap komponen ditampilkan pada Tabel 4.

Tabel 4. Analisis Hasil Implementasi Sistem Keamanan Terintegrasi

Komponen	Hasil Implementasi	Analisis
Suricata IPS	Berhasil mendeteksi dan menjatuhkan lalu lintas SQLi dan <i>Stored XSS</i>	<i>Rule</i> berbasis <i>signature</i> dan pola mampu mengidentifikasi serta menjatuhkan lalu lintas yang sesuai dengan karakteristik serangan
Wazuh SIEM	Berhasil menerima dan mencatat <i>Alert</i> dari <i>Suricata</i>	<i>Log Suricata</i> berhasil diproses dan ditampilkan sebagai <i>event</i> keamanan untuk mendukung analisis terpusat
Wazuh Dashboard	Berhasil menampilkan <i>Alert</i> keamanan dan <i>event</i> respons	Mendukung pemantauan serta penelusuran urutan <i>event</i> deteksi dan respons keamanan secara terpusat
Active Response	Berhasil menjalankan <i>Firewall-drop</i> pada skenario yang memenuhi <i>rule</i>	Mitigasi otomatis dijalankan ketika <i>Rule</i> ID 100005 untuk SQLi atau <i>Rule</i> ID 100060 untuk <i>Stored XSS</i> terpenuhi
Firewall	Berhasil memblokir alamat IP pada skenario yang memicu <i>Active Response</i>	Pemblokiran IP menjadi lapisan mitigasi setelah kondisi <i>rule</i> terpenuhi, sedangkan pada skenario yang tidak memenuhi kondisi <i>Active Response</i> tidak dilakukan pemblokiran

Berdasarkan Tabel 4, Suricata IPS berperan mendeteksi pola serangan melalui signature, sedangkan Wazuh SIEM memproses log untuk pemantauan dan analisis terpusat, yang kemudian divisualisasikan melalui Wazuh Dashboard untuk menelusuri proses deteksi hingga mitigasi. Dalam implementasinya, Rule ID 100005 untuk SQLi dan Rule ID 100060 untuk Stored XSS menggunakan severity level 12 sebagai pemicu Active Response, sehingga keduanya merupakan rule khusus respons, bukan penanda seluruh Alert serangan. Hasil pengujian juga menunjukkan Rule ID 86601 sebagai Alert Suricata dan Rule ID 31106 sebagai peristiwa HTTP dengan severity level 6, sedangkan pada skenario Authentication Bypass SQLi, Alert yang diterima Wazuh memiliki severity level 3. Dengan demikian, severity level bergantung pada jenis peristiwa dan kondisi rule yang terpenuhi. Apabila Rule ID 100005 atau 100060 terpenuhi berdasarkan korelasi peristiwa pada log Apache, Wazuh menjalankan Firewall-drop sebagai Active Response. Mekanisme ini menunjukkan keterkaitan antara signature Suricata sebagai sumber informasi deteksi, proses korelasi Wazuh, Rule ID, severity level, dan tindakan respons.

Signature Suricata disesuaikan dengan karakteristik tiap skenario, yaitu ET WEB_SERVER Possible SQLi Attempt UNION SELECT (SID 2006447) dan signature kustom SQLi Detected untuk SQLi, serta SID 1000010 dan 1000011 untuk mendeteksi pola skrip pada URL dan request body Stored XSS. Signature menghasilkan Alert yang diteruskan ke Wazuh Dashboard, dengan tindak lanjut ditentukan oleh rule yang terpenuhi. Pada Error-Based dan Union-Based SQLi, Alert Suricata (Rule ID 86601) memenuhi Rule ID 100005 (severity level 12) sehingga Active Response dijalankan; mekanisme serupa berlaku pada Stored XSS melalui Rule ID 100060 dengan severity level yang sama. Sebaliknya, Authentication Bypass SQLi tetap terdeteksi Suricata, tetapi tidak memenuhi Rule ID 100005 sehingga IP sumber tidak diblokir. Dengan demikian, Suricata berperan pada deteksi awal, sedangkan Wazuh menentukan klasifikasi peristiwa dan eksekusi respons.

Pada tahap mitigasi, Firewall-drop dijalankan hanya setelah kondisi Active Response terpenuhi, sesuai konfigurasi `<command>Firewall-drop</command>`, `<location>local</location>`, `<rule_id>100005,100060</rule_id>`, dan `<timeout>0</timeout>` yang masing-masing menentukan jenis tindakan, lokasi eksekusi, aturan pemicu, dan sifat blokir permanen tanpa batas waktu otomatis. Alur ini konsisten dengan Gambar 4: lalu lintas diperiksa Suricata hasil deteksi diteruskan Wazuh Agent ke Wazuh Manager kemudian dievaluasi terhadap Rule ID 100005/100060. Jika kondisi terpenuhi, Wazuh mengeksekusi Firewall-drop dan memblokir IP sumber (terbukti pada aturan DROP terhadap 192.168.100.12); jika tidak, Alert hanya dicatat pada Wazuh Dashboard tanpa pemblokiran. Sebagai sumber log, Wazuh Agent menggunakan fast.log (format sederhana untuk verifikasi manual) dan eve.json (data terstruktur berisi waktu kejadian, IP sumber/tujuan, signature ID, nama signature, kategori, dan severity level) sebagai mana ditunjukkan pada Gambar 11.

Penelitian terdahulu menunjukkan bahwa Suricata dapat digunakan untuk mendeteksi dan mencatat serangan serta mendukung pemantauan Alert melalui integrasi dengan berbagai sistem keamanan. Integrasi Suricata dengan sistem pemantauan dan respons keamanan juga telah diterapkan untuk mendukung penanganan serangan pada tingkat jaringan. Dengan demikian, hasil penelitian menunjukkan bahwa integrasi Suricata IPS, Wazuh SIEM, Active Response, dan Firewall mendukung proses deteksi, pemantauan, serta mitigasi serangan aplikasi web berdasarkan kondisi rule yang telah ditetapkan. Hasil pengujian pada tiga skenario yang berhasil dimitigasi memperlihatkan kesesuaian antara Alert Suricata, event Active Response, dan aturan DROP pada Firewall. Pada Authentication Bypass SQLi, tidak adanya pemblokiran menunjukkan bahwa mekanisme mitigasi bekerja berdasarkan hasil deteksi dan kondisi rule, bukan melakukan pemblokiran terhadap seluruh aktivitas pengujian secara indiscriminatif.

5. Kesimpulan

Hasil penelitian menunjukkan bahwa integrasi Suricata sebagai IPS, Wazuh SIEM, dan *Firewall* mendukung deteksi serta mitigasi serangan melalui mekanisme *Active Response*. Dari 4 skenario pengujian, 3 skenario, yaitu *Error-Based SQLi*, *Union-Based SQLi*, dan *Stored XSS*, berhasil dideteksi, direspons, dan diblokir, sedangkan *Authentication Bypass SQLi* tidak mencapai tahap eksploitasi. Pada *Error-Based SQLi*, Suricata mencatat 7 *event*, Wazuh mencatat 4 *Alert* Suricata, 5 *Active Response*, dan 5 *Firewall-drop*, sedangkan *Firewall* mencatat 281 paket dan 19.415 bytes. Pada *Union-Based SQLi*, Suricata mencatat 4 *Alert*, Wazuh mencatat 6 kejadian dengan 4 *Alert* Suricata, 1 *Active Response*, dan 1 *Firewall-drop*, dengan *Firewall* mencatat 215 paket dan 15.814 bytes. Pada *Stored XSS*, Suricata mencatat 4 *Alert*, Wazuh mencatat 4 *Alert* Suricata, 1 *Active Response*, dan 5 *Firewall-drop*, sedangkan *Firewall* mencatat 73 paket dan 8.329 bytes. Sementara itu, *Authentication Bypass SQLi* menghasilkan 46 *event* Suricata dan 15 *Alert* Wazuh, tetapi tidak memicu *Active Response* maupun pemblokiran *Firewall*. Dengan demikian, 3 dari 4 skenario (75%) memenuhi kondisi *Active Response* dan menghasilkan pemblokiran IP, sedangkan seluruh aktivitas yang terdeteksi dapat dipantau melalui Wazuh *Dashboard*.

Penelitian selanjutnya dapat memperluas pengujian dengan menambahkan serangan aplikasi *web* seperti *Command Injection*, *File Inclusion*, dan *Cross-Site Request Forgery* (CSRF) untuk mengevaluasi kemampuan sistem terhadap karakteristik serangan yang lebih beragam. *Rule* Suricata dan Wazuh juga dapat dikembangkan untuk meningkatkan spesifisitas deteksi dan mengurangi false positive. Evaluasi selanjutnya dapat mencakup waktu deteksi, waktu respons, waktu pemblokiran, serta dampak *Active Response* terhadap kinerja layanan *web*. Selain itu, pemanfaatan EVE JSON secara lebih mendalam dapat memperkaya korelasi *event* dan mendukung analisis forensik keamanan.

Pernyataan Penggunaan Artificial Intelligence. Penulis menggunakan ChatGPT untuk membantu pemeriksaan dan perbaikan tata bahasa, pemeriksaan kesalahan pengetikan (typo), penerjemahan abstrak, serta peningkatan kejelasan naskah. Seluruh keluaran yang dihasilkan oleh AI telah diperiksa dan diverifikasi oleh penulis. Penulis bertanggung jawab sepenuhnya atas keakuratan, orisinalitas, dan integritas isi artikel.

Pernyataan Kontribusi Penulis. **A** berkontribusi dalam konseptualisasi, metodologi, pengumpulan data, pengembangan sistem, dan analisis penelitian. **RA** berkontribusi dalam supervisi, pemberian masukan terhadap metodologi, serta peninjauan hasil penelitian. **LW** berkontribusi dalam supervisi, peninjauan, dan penyuntingan naskah. Seluruh penulis telah membaca, meninjau, dan menyetujui versi akhir artikel serta bertanggung jawab atas seluruh aspek penelitian dan publikasi.

Pernyataan Konflik Kepentingan. Penulis menyatakan bahwa tidak terdapat konflik kepentingan yang berkaitan dengan penelitian, kepengarangan, dan publikasi artikel ini.

Pernyataan Pendanaan. Penelitian ini tidak menerima pendanaan khusus dari lembaga pemerintah, komersial, maupun organisasi nirlaba.

Ucapan Terima Kasih. Penulis mengucapkan terima kasih kepada Program Studi Ilmu Komputer, Universitas Bumigora, atas dukungan fasilitas dan lingkungan akademik yang mendukung pelaksanaan penelitian ini. Penulis juga menyampaikan apresiasi kepada seluruh pihak yang telah memberikan bantuan tek-

nis, masukan, dan dukungan selama proses implementasi, pengujian, serta penyusunan penelitian ini.

Daftar Pustaka

- [1] S. S. Nasim, P. Pranav, and S. Dutta, "A systematic literature review on intrusion detection techniques in cloud computing," *Discover Computing*, vol. 28, no. 1, pp. 2–35, 2025. <https://doi.org/10.1007/s10791-025-09641-y>.
- [2] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, no. 102419, pp. 1–20, 2020. <https://doi.org/10.1016/j.jisa.2019.102419>.
- [3] C. Moharir, S. K. Lingishetty, and A. Kamboj, "Securing web applications against sql injection and xss attacks," *Journal of Information Security and Applications*, vol. 14, no. 5, pp. 203–208, 2025. <https://doi.org/10.51583/IJLTEMAS.2025.140500025>.
- [4] S. E. Prasetyo, H. Haeruddin, and K. Ariesryo, "Sistem keamanan website dari serangan denial of service, sql injection, cross site scripting menggunakan web application firewall," *Antivirus: Jurnal Ilmiah Teknik Informatika*, vol. 18, no. 1, pp. 27–36, 2024. <https://doi.org/10.35457/antivirus.v18i1.3339>.
- [5] C. Chamberland, "2024 annual wordpress vulnerability and threat report by wordfence," Apr. 2025. Accessed: Sep. 18, 2026. [Online]. Available: <https://www.wordfence.com/wp-content/uploads/2025/04/2024-Annual-WordPress-Security-Report-by-Wordfence.pdf>.
- [6] A. Paul, V. Sharma, and O. Olukoya, "Sql injection attack: Detection, prioritization & prevention," *Journal of Information Security and Applications*, vol. 85, no. 103871, pp. 1–21, 2024. <https://doi.org/10.1016/j.jisa.2024.103871>.
- [7] G. Deshmukh, R. Kamble, and P. S. Solanki, "Protection of network devices and data security using firewall: A literature survey," *International Journal of Computer Sciences and Engineering*, vol. 9, no. 11, pp. 39–44, 2021. <https://doi.org/10.26438/ijcse/v9i11.3944>.
- [8] P. Senthil, B. P. Kavin, S. R. Srividhya, V. Ramachandran, C. Kavitha, and W. C. Lai, "Performance evaluation of stateful firewall-enabled sdn with flow-based scheduling for distributed controllers," *Electronics (Switzerland)*, vol. 11, no. 19, pp. 2–20, 2022. <https://doi.org/10.3390/electronics11193000>.
- [9] E. Kristi, A. Tobing, R. E. Septya, and Y. Servanda, "Comparative analysis of network security: Firewall, ids, and ai-based defense against ddos attacks," *Journal of Artificial Intelligence and Engineering Applications (JAIEA)*, vol. 4, no. 3, pp. 1818–1822, 2025. <https://doi.org/10.59934/jaiea.v4i3.1026>.
- [10] E. Stephani, F. Nova, E. Asri, and N. Fitri, "Implementasi dan analisa keamanan jaringan ids (intrusion detection system) menggunakan suricata pada web server," *JITSI: Jurnal Ilmiah Teknologi Sistem Informasi*, vol. 1, no. 2, pp. 67–74, 2020. <https://doi.org/10.62527/jitsi.1.2.10>.
- [11] F. T. Anugrah *et al.*, "Implementasi intrusion prevention system (ips) menggunakan suricata untuk serangan sql injection," *Techné: Jurnal Ilmiah Elektroteknika*, vol. 21, no. 2, pp. 199–210, 2022. <https://doi.org/10.31358/techné.v21i2.320>.
- [12] H. Asad and I. Gashi, "Dynamical analysis of diversity in rule-based open source network intrusion detection systems," *Empirical Software Engineering*, vol. 27, no. 1, pp. 2–30, 2022. <https://doi.org/10.1007/s10664-021-10046-w>.
- [13] M. A. Mishra, "Integration of siem for real-time threat detection," *International Journal of Engineering Development and Research (IJEDR)*, vol. 14, no. 1, pp. 512–520, 2026. <https://rjwave.org/ijedr/papers/IJEDR2601447.pdf>.
- [14] A. Alanda, H. A. Mooduto, and R. Hadi, "Real-time defense against cyber threats: Analyzing wazuh's effectiveness in server monitoring," *JITCE*, vol. 7, no. 2, pp. 56–62, 2023. <https://doi.org/10.25077/jitce.7.2.56-62.2023>.
- [15] H. A. Damanik and M. Anggraeni, "Sistem deteksi intrusi hybrid dan mitigasi kerentanan infrastruktur jaringan menggunakan teknik active response (xdr) wazuh dan suricata," *Jurnal Pekommas*, vol. 9, no. 2, pp. 309–322, 2024. <https://doi.org/10.56873/jpkm.v9i2.5829>.
- [16] R. S. Wiandani, M. Tahir, I. A. Dyransya, and R. Ummah, "Identifikasi serangan sql injection berbantuan aplikasi pengujian keamanan web dvwa (damn vulnerable web application)," *Digital Transformation Technology (Digitech)*, vol. 5, no. 1, pp. 375–382, 2025. <https://doi.org/10.47709/digitech.v5i1.5922>.
- [17] H. Setiawan, W. Sulisty, F. T. Informasi, and U. K. S. Wacana, "Siem (security information event management) model for malware attack detection using suricata and evebox," *International Journal of Engineering Technology and Natural Sciences (IJETS)*, vol. 5, no. 2, pp. 138–147, 2023. <https://doi.org/10.46923/ijets.v5i2.241>.

- [18] R. Rodianto, I. Idham, Y. Yuliadi, M. T. A. Zaen, and W. Ramadhan, “Penerapan network development life cycle (ndlc) dalam pengembangan jaringan komputer pada badan pengelolaan keuangan dan aset daerah (bpkad) provinsi ntb,” *Jurnal Ilmiah FIFO*, vol. 14, no. 1, pp. 35–46, 2022. <http://dx.doi.org/10.22441/fifo.2022.v14i1.004>.
- [19] M. Djamalyanto, L. Widyawati, and I. P. Hariyadi, “Implementasi security information and event management untuk mencegah serangan deface pada server terintegrasi telegram,” *Melek IT: Information Technology Journal*, vol. 11, no. 1, pp. 31–42, 2025. <https://doi.org/10.30742/melekitjournal.v11i1.398>.