JUPITER
Teknologi Informatika & Komputer
Vol. 6 No. 1 Februari 2025
Institut Bisnis Muhammadiyah Bekasi
P-ISSN: 2798-3129 , E-ISSN: 2798-6187

A SIEM-DRIVEN SOLUTION FOR CYBER ATTACK DETECTION IN EDUCATIONAL *WEBSITE*S: IMPLEMENTING *THREAT LOG FILTERING* FOR ENHANCED SECURITY

Sutisno Candra Kusuma[1], Yunus Fadhillah Soleman[1,*], Budi Berlinton[2], Jamaludin[1] Supriyanto Karya[3]

[1]Teknik Informatika, Institut Bisnis Muhammadiyah, sutisnocandrakusuma@gmail.com, yunus@ibm.ac.id , jamalthea007@ibm.ac.id
[2]Teknik Informatika, Universitas Multimedia Nusantara, budi.sitorus@umn.ac.id
[3]Informatika, Universitas Indonesia Mandiri, supriyantokarya.sk@outlook.com

ABSTRAK

Penelitian bertujuan untuk meningkatkan deteksi serangan siber dengan implementasi *Security Information and Event Management (SIEM)* menggunakan metode *Threat Log Filtering*. Data *log* dikumpulkan dari berbagai sumber, termasuk server dan jaringan, lalu dianalisis untuk mengidentifikasi aktivitas mencurigakan. Hasil penelitian menunjukkan bahwa metode ini mampu meningkatkan kecepatan deteksi serangan hingga 15% dan mengurangi tingkat false positive sebesar 5%. Implementasi ini berkontribusi pada peningkatan keamanan siber organisasi dengan memitigasi ancaman sebelum berdampak luas.
Kata Kunci: *Deteksi Intrusi, Keamanan Siber, SIEM, Suricata, Threat Log Filtering, Wazuh*

ABSRACT

This study aims to improve cyber attack detection by implementing Security Information and Event Management (SIEM) using the Threat Log Filtering method. Log data is collected from various sources, including servers and networks, and then analyzed to identify suspicious activity. The results of the study show that this method is able to increase the speed of attack detection by up to 15% and reduce the false positive rate by 5%. This implementation contributes to improving the organization's cybersecurity by mitigating threats before they have a wide impact.
Keywords: *Cyber Security, Intrusion Detection, SIEM, Suricata, Threat Log Filtering, Wazuh*

PENDAHULUAN

Pentingnya keamanan sebuah *website* dalam institusi pendidikan sangatlah penting karena pada *website* tersebut menyimpan informasi sensitif tentang siswa dan staf, termasuk data pribadi, catatan akademis, dan materi pengajaran. Menurut laporan Badan Siber dan Sandi Negara (BSSN, 2023) terjadi 370,02 Juta serangan siber pada tahun 2022, terdapat peningkatan serangan siber sebesar 38,72% pada tahun sebelumnya meskipun sektor pendidikan tidak ada data detilnya. Beberapa penelitian sebelumnya telah mengeksplorasi penggunaan SIEM dalam deteksi serangan, seperti penelitian oleh Bezas (2023) yang membandingkan efektivitas berbagai sistem SIEM open-source. Namun, penelitian ini berfokus pada optimalisasi deteksi dengan metode *Threat Log Filtering* dan mengintegrasikannya dengan *Suricata* dan *Wazuh* untuk meningkatkan respons terhadap serangan

Institut Bisnis Muhammadiyah Bekasi mengandalkan *LiteSpeed Web Server* untuk operasional *website*-nya. Namun, sistem ini lebih berfokus pada kecepatan pemrosesan dibandingkan aspek keamanan. Akibatnya, *website* masih rentan terhadap serangan siber. Oleh karena itu, diperlukan solusi yang dapat meningkatkan deteksi dan respons terhadap serangan secara real-time. Salah satu pendekatan yang dapat diterapkan adalah Security

Information and Event Management (SIEM), yang mampu mengumpulkan dan menganalisis *log* dari berbagai sumber untuk mengidentifikasi ancaman potensial. Metode *Threat Log Filtering* digunakan untuk memfilter *log* berdasarkan aturan tertentu guna meningkatkan efisiensi deteksi serangan.

TINJAUAN PUSTAKA

Security Information and Event Management

SIEM adalah sistem keamanan yang menggabungkan manajemen informasi keamanan (SIM) dan manajemen peristiwa keamanan (SEM). SIEM berfungsi mengumpulkan, menyimpan, dan menganalisis data *log* dari berbagai sumber dalam suatu sistem untuk mendeteksi ancaman secara real-time (Caldeira & Capstone, 2021). Metodo*log*i SIEM:

1. Pengumpulan Data – Mengumpulkan *log* dari perangkat jaringan, server, aplikasi, dan endpoint.
2. Analisis dan Korelasi – Data dianalisis menggunakan aturan dan algoritma untuk mendeteksi ancaman.
3. Deteksi Ancaman – Sistem mengenali pola mencurigakan berdasarkan *signature* atau machine learning.
4. Respon Insiden – SIEM dapat memicu peringatan otomatis atau mengambil tindakan mitigasi.

Menurut (Bezas, 2023) Dalam prosesnya *SIEM* memiliki cara kerjanya sendiri. Cara kerja *SIEM* sendiri memiliki beberapa tahapan diantaranya proses pengumpulan data, proses analisa dan standarisasi, melakukan korelasi dan keterkaitan dalam data, proses deteksi ancaman, memberikan peringatan serta respon, penyusunan laporan keamanan, dan analisa serta peningkatan keamanan.

Elasticsearch, Logstash, Kibana Stack

ELK Stack digunakan dalam pengelolaan data *log* untuk mendukung sistem SIEM (Ngo et al., 2021). Penjelasan masing-masing komponen ELK Stack:

1. *Elasticsearch*: *Database* pencarian berbasis RESTful yang memungkinkan pencarian cepat dan analisis data *log* (Gambar 1)..
2. *Logstash*: Alat yang digunakan untuk mengambil, memproses, dan mengirimkan data *log* ke Elasticsearch.
3. Kibana: Antarmuka visualisasi data yang mempermudah analisis *log* melalui *dashboard*.

```
{
  "query": {
    "match": {
      "message": "failed login"
    }
  }
}
```

Sumber : Hasil Penelitian (2024)

Gambar 1. *JSON Query Example*

Query mencari pesan *log* yang berisi "*failed login*" untuk mendeteksi percobaan *log*in yang mencurigakan.

Wazuh

Wazuh adalah suatu *platform* keamanan yang berfokus pada deteksi ancaman dan manajemen keamanan. *Wazuh* menyediakan berbagai fitur termasuk analisis *log*, deteksi intrusi, keberlanjutan pengawasan keamanan, dan respon terhadap insiden. *Wazuh* merupakan solusi *open-source* yang dapat digunakan untuk meningkatkan keamanan sistem dan jaringan dengan mendeteksi perilaku mencurigakan atau serangan siber (Laksmiati, 2021). *Wazuh* menyediakan fitur *XDR (Extended Detection and Response)* dan *SIEM* untuk melindungi beban kerja *cloud*, *container*, dan *server*, *wazuh* didasarkan pada agent *Wazuh*, yang disebarkan pada titik akhir yang dipantau, dan pada tiga komponen utama yaitu:

1. *Wazuh indexer* adalah mesin analitik dan pencarian teks lengkap yang sangat skalabel.
2. *Wazuh server* menganalisis data yang diterima dari agen. Komponen ini memprosesnya melalui decoder dan *rules*, menggunakan threat intelligence untuk mencari *indicators of compromise (IOCs).*
3. *Wazuh dashboard* adalah antarmuka pengguna *web* untuk visualisasi dan analisis data.

Suricata

Suricata adalah sistem deteksi intrusi dan pencegahan serangan *(IDS/IPS) open-source* yang digunakan untuk memantau lalu lintas jaringan dan mendeteksi potensi serangan siber. *Suricata* berfokus pada mendeteksi dan merespons terhadap ancaman di tingkat jaringan dengan menganalisis paket-paket data yang melintas melalui suatu sistem (Murphy, 2019). Deteksi intrusi adalah proses memantau lalu lintas jaringan dan menganalisisnya untuk mencari tanda-tanda kemungkinan intrusi, seperti upaya eksploitasi dan insiden yang mungkin merupakan ancaman terhadap jaringan (Ernawati et al., 2019). Pencegahan intrusi adalah proses melakukan deteksi intrusi dan kemudian menghentikan insiden yang terdeteksi, biasanya dilakukan dengan menjatuhkan paket atau mengakhiri sesi.

Telegram

Telegram adalah *platform* pesan instan yang didirikan oleh Pavel Durov pada tahun 2013. *Platform* ini dirancang untuk menyediakan layanan komunikasi yang aman, cepat, dan terenkripsi *end-to-end*. *Telegram* memungkinkan pengguna untuk mengirim pesan teks, gambar, audio, video, dan berbagai jenis file kepada kontak atau dalam grup tertentu. *Telegram* juga menyediakan berbagai fitur lain, termasuk saluran (*channel*), grup, dan *bot* yang dapat diprogram untuk melakukan berbagai tugas otomatis. Menurut (Thomas & Bhat, 2022), *Telegram* diakui karena implementasi enkripsinya yang kuat dan fokus pada keamanan. *Telegram bot* adalah program otomatis yang diintegrasikan dengan *platform Telegram* menggunakan *Telegram Bot API*. *Bot* ini dapat digunakan untuk melakukan berbagai tugas dan memberikan respons otomatis kepada pengguna (Mulyanto, 2020).

Telegram API adalah antarmuka pemrograman aplikasi yang diberikan oleh *Telegram Messenger*. *Telegram API* memungkinkan pengembang untuk terhubung dan berinteraksi dengan layanan *Telegram*, memanfaatkan berbagai fungsi dan fitur yang disediakan oleh *platform* tersebut. Menurut penelitian yang telah dilakukan oleh (Nufusula & Susanto, 2023), *Telegram API* menyediakan serangkaian metode pemrograman yang memungkinkan pengembang untuk membuat aplikasi pihak ketiga

yang terintegrasi dengan *Telegram*. Telegram API digunakan untuk mengirim notifikasi otomatis dari sistem SIEM. Bot Telegram digunakan untuk berinteraksi dengan sistem keamanan dengan menggunakan metode HTTP *request* (Gambar 2). Perintah yang digunakan untuk mengirim pesan peringatan ke Telegram.

```
https://api.telegram.org/bot<TOKEN>/sendMessage?chat_id=<ID>&text=Alert
```

Sumber : Hasil Penelitian (2024)

Gambar 2. Metode *HTTP Request*

Python

Python merupakan bahasa pemrograman yang bersifat serbaguna dan mudah dipahami, sangat cocok untuk berbagai keperluan pengembangan perangkat lunak. Dalam penelitian tentang bahasa pemrograman modern yang ditulis oleh (Jalolov, 2023). Python digunakan dalam otomatisasi keamanan dan analisis *log* (Gambar 3).

```python
import re
log = "Failed login from 192.168.1.1"
if re.search("Failed login", log):
    print("Alert: Possible brute-force attack!")
```

Sumber : Hasil Penelitian (2024)

Gambar 3. Kode Deteksi Serangan dengan *Python*

Serangan Siber

Serangan siber (*cyber attack*) merupakan sebuah serangan yang dilakukan oleh individu, kelompok, organisasi maupun negara. Tujuan serangan siber itu sendiri adalah mencuri, mengubah, merusak yang dapat merugikan organisasi atau individu yang menjadi target serangan. Serangan siber memiliki banyak macam diantaranya adalah Reconnaissance (Pengumpulan Informasi), *Brute Force*, *Command Injection*, *SQL Injection*, dan *Cross-Site Scripting*.

Standarisasi Keamanan

ISO/IEC 27001:2022 adalah versi terbaru dari standar internasional yang menetapkan persyaratan untuk Sistem Manajemen Keamanan Informasi *(Information Security Management System* atau *ISMS*) (Legowo & Juhartoyo, 2022). Elemen Utama ISO 27001:

1. *Risk Assessment* – Identifikasi ancaman terhadap informasi.
2. *Access Control* – Membatasi akses terhadap data sensitif.
3. *Incident Management* – Menentukan prosedur respons insiden keamanan.

Standar tersebut dirancang untuk memastikan bahwa informasi di seluruh organisasi dilindungi dari ancaman seperti *cyber attack*, pencurian data, atau kerusakan sistem. ISO 27001:2022 merupakan pembaruan dari ISO 27001:2013, pembaruan yang terjadi adalah terdapat 93 kontrol yang ada pada ISO 27001:2022 ini termasuk pada 11 kontrol baru, 24 penggabungan beberapa kontrol, dan 58 kontrol yang diperbaharui.

Log Management

Log merupakan sebuah catatan sistem atau catatan kejadian yang dibuat oleh sistem komputer atau perangkat lunak. *Log* ini mencatat berbagai informasi, seperti peristiwa keamanan, aktivitas pengguna, kesalahan sistem, atau peristiwa lainnya yang dapat digunakan untuk pemantauan, audit, dan analisis keamanan (Anastopoulos & Katsikas, 2019). *Log Management* merujuk pada proses pengumpulan, penyimpanan, analisis, retensi, dan pemantauan *log* dari berbagai sumber dalam suatu sistem atau jaringan. Tujuan utama dari *log management* adalah untuk mendukung kebutuhan keamanan informasi, pemantauan kesehatan sistem, dan kepatuhan regulasi dengan menyediakan rekam jejak yang lengkap dari peristiwa yang terjadi di dalam suatu organisasi (Pratama D et al., 2022).

Manajemen *log* mencakup pengumpulan, penyimpanan, dan analisis data *log* untuk mendukung audit keamanan. Rumus Retensi Log (Rumus 1). Jika volume *log* harian 5GB dan retensi 30 hari adalah Total Storage = 5GB x 30 sehingga 150GB. Penerapan strategi penyimpanan *log* yang optimal memastikan ketersediaan data untuk analisis keamanan.

$$Total\ Storage = (Daily\ Log\ Volume)\ x\ (Retention\ Period) \quad (1)$$

Penelitian Relevan

Bezas (2003) melakukan analisis komparatif terhadap berbagai sistem Security Information and Event Management (SIEM) berbasis open-source. Penelitian ini menyoroti efektivitas SIEM dalam mendeteksi serangan siber dan membandingkan fitur-fitur yang dimiliki oleh beberapa sistem SIEM populer seperti OSSIM, *Wazuh*, dan *Suricata*. Hasil penelitian menunjukkan bahwa OSSIM merupakan salah satu sistem SIEM open-source yang paling lengkap dan efektif dalam mendeteksi serangan siber. Penelitian ini relevan dengan studi ini karena memberikan gambaran tentang kelebihan dan kekurangan berbagai sistem SIEM yang dapat diintegrasikan dengan metode *Threat Log Filtering*.

Penelitian Harahap, A. G. S., & Hutrianto (2023) menguji efektivitas *Wazuh* sebagai alat audit dan deteksi serangan siber. Hasil penelitian menunjukkan bahwa *Wazuh* mampu mendeteksi serangan dengan cepat dan akurat, terutama pada sistem operasi Windows 7 yang memiliki lebih banyak kerentanan dibandingkan sistem operasi lainnya. Penelitian ini relevan karena memberikan bukti empiris tentang kemampuan *Wazuh* dalam mendeteksi serangan siber, yang menjadi salah satu komponen utama dalam implementasi SIEM pada studi ini.

Penelitian Rahmawati, T., Karna, N. B. A., & Hertiana, S. N. (2023) membahas perancangan sistem SIEM menggunakan aplikasi Snorby sebagai antarmuka untuk manajemen Intrusion Detection System (IDS). Hasil penelitian menunjukkan bahwa SIEM dapat mengumpulkan, menganalisis, dan memberikan laporan terkait keamanan data dari berbagai sumber. Penelitian ini relevan karena memberikan gambaran tentang bagaimana SIEM dapat diintegrasikan dengan sistem deteksi intrusi untuk meningkatkan keamanan jaringan.

Penelitian Caldeira, H. (2021) membahas implementasi SIEM untuk meningkatkan keamanan jaringan dengan memulai respons otomatis terhadap serangan siber. Caldeira menekankan pentingnya optimalisasi SIEM untuk mengurangi risiko pelanggaran keamanan. Penelitian ini relevan karena memberikan rekomendasi tentang

bagaimana SIEM dapat digunakan untuk memblokir serangan secara otomatis, yang sejalan dengan tujuan studi ini dalam mengimplementasikan *Threat Log Filtering*.

Penelitian Ngo, T. T. T., et al. (2021) membahas penggunaan ELK Stack (Elasticsearch, Logstash, Kibana) dalam pengelolaan data *log* untuk mendukung sistem SIEM. ELK Stack digunakan untuk mengumpulkan, memproses, dan menganalisis data *log* secara real-time. Penelitian ini relevan karena ELK Stack merupakan salah satu komponen penting dalam implementasi SIEM pada studi ini, terutama dalam hal pengelolaan dan analisis *log*.

Penelitian Laksmiati, D. (2021) mengimplementasikan *Wazuh* 4.0 untuk melindungi integritas file dan mendeteksi serangan siber. Hasil penelitian menunjukkan bahwa *Wazuh* efektif dalam mendeteksi perubahan file yang mencurigakan dan memberikan respons cepat terhadap insiden keamanan. Penelitian ini relevan karena *Wazuh* digunakan sebagai salah satu komponen utama dalam sistem SIEM yang diimplementasikan pada studi ini.

Penelitian Mulyanto, A. D. (2020) membahas pemanfaatan *bot Telegram* sebagai media informasi dan notifikasi otomatis. Hasil penelitian menunjukkan bahwa *bot Telegram* efektif dalam memberikan respons cepat terhadap insiden keamanan. Penelitian ini relevan karena *bot Telegram* digunakan dalam studi ini untuk mengirim notifikasi otomatis ketika sistem SIEM mendeteksi serangan siber.

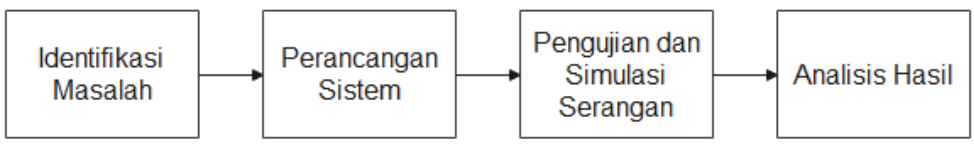
Penelitian Murphy, B. R. (2019) membandingkan kinerja sistem deteksi intrusi Snort dan *Suricata*. Hasil penelitian menunjukkan bahwa *Suricata* memiliki performa yang lebih baik dalam mendeteksi serangan siber, terutama dalam hal kecepatan dan akurasi deteksi. Penelitian ini relevan karena *Suricata* digunakan sebagai sistem deteksi dan pencegahan intrusi (IDPS) dalam studi ini.

Penelitian Ernawati, T., Fachrozi, M. F., & Syaputri, D. D. (2019) menganalisis kinerja sistem deteksi intrusi (IDS) dalam mendeteksi serangan siber. Hasil penelitian menunjukkan bahwa IDS dapat mendeteksi berbagai jenis serangan dengan tingkat akurasi yang tinggi. Penelitian ini relevan karena memberikan gambaran tentang bagaimana sistem deteksi intrusi dapat diintegrasikan dengan SIEM untuk meningkatkan keamanan jaringan.

Penelitian Anastopoulos, V., & Katsikas, S. (2019) mengusulkan metodo*log*i untuk merancang infrastruktur manajemen *log* yang adaptif. Hasil penelitian menunjukkan bahwa manajemen *log* yang efektif dapat meningkatkan kemampuan deteksi ancaman dan mengurangi waktu respons terhadap insiden keamanan. Penelitian ini relevan karena manajemen *log* merupakan komponen kunci dalam implementasi SIEM dan metode *Threat Log Filtering*

METODE PENELITIAN

Pada penelitian terdapat beberapa tahapan proses penelitian yang dibuat peneliti seperti pada Gambar 4.



Sumber : Hasil Penelitian (2024)

Gambar 4. Metode Penelitian

Tahapan peneliti melakukan proses penelitian sebagai berikut:

1. Identifikasi masalah, analisis risiko terhadap infrastruktur keamanan *website* Institut Bisnis Muhammadiyah Bekasi.
2. Perancangan sistem, mengimplementasikan SIEM berbasis *Wazuh* dan *Suricata* dengan fitur *Threat Log Filtering*.
3. Pengujian dan simulasi serangan, melakukan pengujian terhadap X jenis serangan siber (XSS, SQL Injection, Brute Force, dll.) selama Y hari dengan total Z percobaan serangan.
4. Analisis hasil, mengevaluasi efektivitas deteksi dengan metrik accuracy, precision, dan recall, serta membandingkan hasilnya dengan metode konvensional.

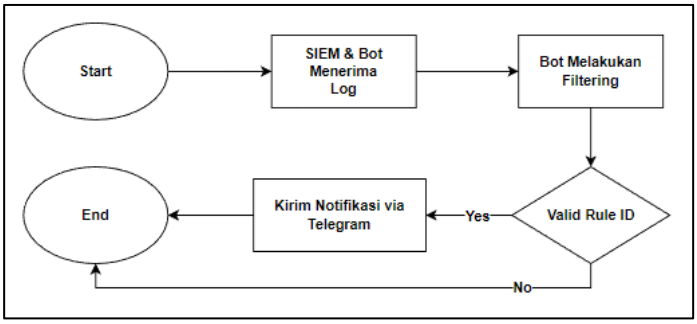
HASIL DAN PEMBAHASAN

Analisis menunjukkan bahwa *website* Institut Bisnis Muhammadiyah Bekasi masih memiliki celah keamanan yang dapat dieksploitasi oleh peretas. Serangan yang terdeteksi meliputi *Cross-Site Scripting (XSS), SQL Injection, Brute Force, dan Command Injection*.

Perancangan Sistem

SIEM berbasis *Wazuh* dan *Suricata* diimplementasikan untuk mengumpulkan *log*, melakukan analisis ancaman, serta mengirimkan notifikasi melalui *bot Telegram*.

1. Konfigurasi SIEM
   SIEM dikonfigurasi menggunakan Ubuntu Server 22.04 sebagai *platform* utama, sementara pengujian serangan dilakukan menggunakan Kali Linux. Sistem memanfaatkan *Wazuh* untuk deteksi intrusi dan *Suricata* sebagai *Intrusion Detection and Prevention System (IDPS).*Sistem Operasi *Kali Linux* 2023.3 digunakan untuk melakukan percobaan penyerangan. Bahasa Pemrograman *Python* digunakan untuk membuat *bot*. *ELK Stack* dan *Wazuh* berperan sebagai *SIEM* yang bertugas sebagai tools monitoring. *Suricata* digunakan sebagai *IDPS (Intrusion Detection Prevention System)*. *Telegram* digunakan untuk mengirimkan notifikasi ancaman keamanan. *Nmap* digunakan sebagai *tools* percobaan serangan. *SQL Map* digunakan sebagai *tools* percobaan serangan.
2. Konfigurasi Bot
   Peneliti merancang *bot* dengan algoritma yang dimulai ketika *SIEM* menerima *log* dan melakukan proses penyaringan atau *filtering* untuk mengetahui apakah *log* tersebut sesuai dengan aturan yang telah ditetapkan (Gambar 5).



Sumber : Hasil Penelitian (2024)

Gambar 5. Algoritma Bot

JUPITER
Teknologi Informatika & Komputer
Vol. 6 No. 1 Februari 2025
Institut Bisnis Muhammadiyah Bekasi
P-ISSN: 2798-3129 , E-ISSN: 2798-6187

Algoritma *bot* menggunakan metode *Threat Log Filtering* yang bekerja jika *log* memiliki Rule ID yang sah, *bot* akan mengirimkan notifikasi melalui Telegram, tetapi jika tidak, *bot* tidak akan mengirimkan notifikasi.

3. *Threat Log Filtering*

Peneliti menulis script menggunakan bahasa pemrograman *python* dan *bash* untuk mengontrol *bot* tersebut dan menuliskan *rule_id* yang valid dalam bentuk *JSON*.

a. *Custom-telegram.py*

Pada *script* Gambar 6 ditulis menggunakan bahasa pemrograman *python* yang berfungsi untuk memproses *alert*, melakukan *filtering rule_id*, membentuk pesan yang valid, dan mengirimkan pesan menuju *telegram* (Gambar 7).

```
# Check if the rule ID is valid
rule_id = alert_json['rule']['id'] if 'rule' in alert_json else ''
if rule_id not in valid_rule_ids:
    print(f"Rule ID {rule_id} is not valid. Alert will not be sent.")
    sys.exit(0)

# Send the request if the rule ID is valid
msg_data = create_message(alert_json)
headers = {'content-type': 'application/json', 'Accept-Charset': 'UTF-8'}
response = requests.post(hook_url, headers=headers, data=msg_data)
```

Sumber : Hasil Penelitian (2024)

Gambar 6. Proses Seleksi Rule ID

b. *Custom-telegram.sh*

Script tersebut ditulis menggunakan bahasa pemrograman *bash* yang berfungsi untuk mengeksekusi sebuah *script python* yang lokasinya ditentukan berdasarkan lokasi dari *script shell* yang sedang dijalankan. *Script* tersebut juga menyesuaikan path berdasarkan struktur *directory* di mana *script shell* berada, sehingga dapat menyesuaikan dengan berbagai lingkungan (Gambar 7).

Sumber : Hasil Penelitian (2024)

Gambar 7. Penetapan Path *Wazuh* dan *Script*

c. *Valid_rule_id.json*

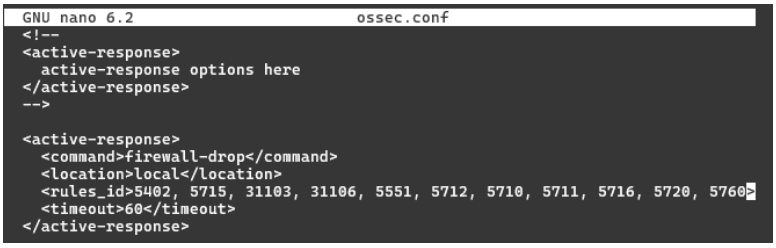
File tersebut berisikan berbagai *rule_id* (Gambar 8) yang memang menandakan adanya sebuah serangan yang valid.

```
{
    "valid_rule_id": ["2502", "5402", "5404", "5503", "5504", "5551",
"5710", "5711", "5712", "5715", "5716", "5720", "5760", "31103",
"31105", "31106", "31154", "40112", "60122", "60204", "86601",
"100201"]
}
```

Sumber : Hasil Penelitian (2024)

Gambar 8. Valid Rule ID

4. Konfigurasi Rules
   Peneliti akan membuat sebuah *custom rules* untuk memblokir serangan sebelumnya menggunakan wazuh active response dengan mengidentifikasi serangannya menggunakan *rule_id* sehingga dapat mencegah adanya serangan yang sama seperti sebelumnya.



Sumber : Hasil Penelitian (2024)

Gambar 9. Konfigurasi Rules

Penelitian *custom rules* ditulis pada *file ossec.conf* (Gambar 6) diletakkan pada bagian *active-response* yang dimana *wazuh* akan melakukan blok apabila ada *alert* atau serangan dengan *rule* yang sudah ditetapkan selama 60 detik.

Pengujian dan Simulasi Penyerangan

1. *Scanning Testing*
   Penyerangan ini dilakukan dengan mencoba melakukan pemindaian menggunakan *tools Nmap*. Ketika peneliti menjalankan *Nmap*, peneliti menggunakan parameter *-Pn* untuk menargetkan *service* yang aktif, *-sC* untuk menjalankan pemeriksaan keamanan termasuk kerentanan, dan *--min-rate 1000* untuk mengatur kecepatan menjadi 1000 paket per detik. Setelah dilakukan 2 kali (Tabel 1 dan Tabel 2) penyerangan terlihat beberapa *port* yang terbuka untuk publik.

Tabel 1. Hasil *Testing* 1 *Scanning*

Port	***Service***
21	*FTP*
22	*SSH*
53	*DNS*
80	*HTTP*

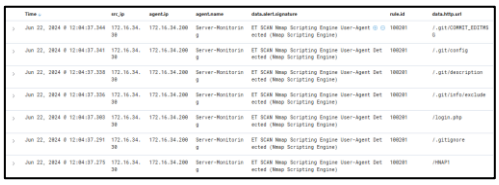
Sumber: Hasil Penelitian (2024)

Tabel 2. Hasil *Testing 2 Scanning*

Port	***Service***
21	*FTP*
22	*SSH*
53	*DNS*
80	*HTTP*
2222	*EthernetIP*

Sumber : Hasil Penelitian (2024)

SIEM berhasil mendeteksi sejumlah besar upaya pemindaian *port* yang mencurigakan yang menandakan adanya sebuah serangan scanning dan juga *SIEM* mendeteksi serangan tersebut dengan nama *"Nmap Scripting Engine Detected"* (Gambar 10).



Sumber : Hasil Penelitian (2024)

Gambar 10. Log Scanning Attack

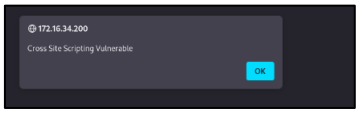
Selain *SIEM* yang dapat mendeteksi *bot* juga berhasil mengirimkan sebuah notifikasi terjadinya sebuah *scanning attack* dengan menampilkan judul, *source IP*, *destination IP*, *rule ID*, *Agent*, teknik dan taktik penyerangan serta *full log* yang menampilkan keseluruhan *log* yang diterima (Gambar 11).

Sumber : Hasil Penelitian (2024)

Gambar 11. Notifikasi *Bot Scanning Attack*

2. *Cross-Site Scripting Testing*

Penyerangan ini dilakukan dengan mencoba melakukan serangan *XSS (Cross-Site Scripting)* dengan menyuntikkan *script* berbahaya ke dalam input. Peneliti menuliskan code *<script>alert("Cross Site Scripting Vulnerabe")</script>* yang merupakan code javascript untuk menampilkan sebuah *pop up* sebuah *alert* atau peringatan. Ketika peneliti menyuntikkan *script* tersebut, terlihat bahwa *script* tersebut dieksekusi oleh *browser* (Gambar 12).



Sumber : Hasil Penelitian (2024)

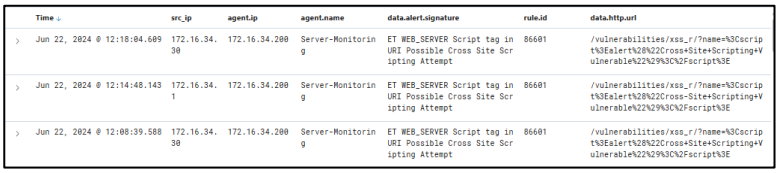
Gambar 12. Testing 1 XSS

Pada Gambar 13 peneliti melakukan penyerangan yang kedua, peneliti menggunakan *script* yang sama pada percobaan pertama terlihat bahwa *script* tersebut masih dieksekusi oleh *browser* namun *browser* tidak menampilkan hasil apapun sehingga penyerangan metode ini sudah berhasil terblokir.

Sumber : Hasil Penelitian (2024)

Gambar 13. Testing 2 XSS

SIEM berhasil mendeteksi adanya proses penyuntikan *script* yang ditulis menggunakan *javascript* ke dalam input yang mengindikasikan adanya sebuah serangan *Cross-Site Scripting (XSS)* dan juga *SIEM* mendeteksi serangan tersebut dengan nama *"Script tag in URI Possible Cross Site Scripting Attemp"* (Gambar 14).

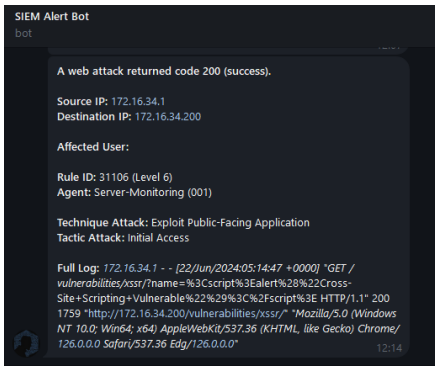


Time	src_ip	agent.ip	agent.name	data.alert.signature	rule.id	data.http.url
Jun 22, 2024 @ 12:18:04.609	172.16.34.30	172.16.34.200	Server-Monitoring	ET WEB_SERVER Script tag in URI Possible Cross Site Scripting Attempt	86601	/vulnerabilities/xss_r/?name=%3Cscript%3Ealert%28%22Cross-Site+Scripting+Vulnerable%22%29%3C%2Fscript%3E
Jun 22, 2024 @ 12:14:48.143	172.16.34.1	172.16.34.200	Server-Monitoring	ET WEB_SERVER Script tag in URI Possible Cross Site Scripting Attempt	86601	/vulnerabilities/xss_r/?name=%3Cscript%3Ealert%28%22Cross-Site+Scripting+Vulnerable%22%29%3C%2Fscript%3E
Jun 22, 2024 @ 12:08:39.588	172.16.34.30	172.16.34.200	Server-Monitoring	ET WEB_SERVER Script tag in URI Possible Cross Site Scripting Attempt	86601	/vulnerabilities/xss_r/?name=%3Cscript%3Ealert%28%22Cross-Site+Scripting+Vulnerable%22%29%3C%2Fscript%3E

Sumber : Hasil Penelitian (2024)

Gambar 14. Log XSS Attack

Selain *SIEM* yang dapat mendeteksi *bot* juga berhasil mengirimkan sebuah notifikasi terjadinya sebuah *Cross-Site Scripting attack* dengan menampilkan judul, *source IP*, *destination IP*, *rule ID*, *Agent*, teknik dan taktik penyerangan serta *full log* yang menampilkan keseluruhan *log* yang diterima (Gambar 15).

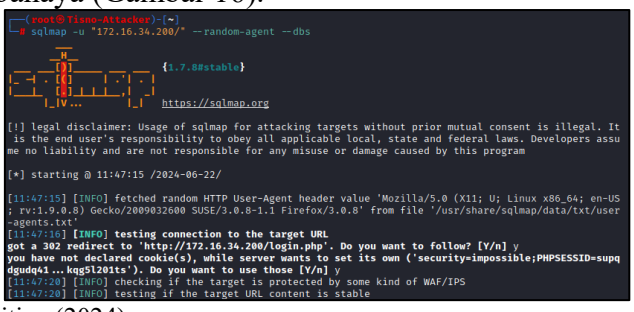


Sumber : Hasil Penelitian (2024)

Gambar 15. Notifikasi Bot XSS Attack

3. *SQL Injection Testing*

Penyerangan ini dilakukan dengan serangan *SQL Injection* menggunakan *tools sqlmap*. Ketika menjalankan *sqlmap*, peneliti menggunakan *parameter --random-agent* yang berguna untuk menggunakan berbagai macam agent browser, dan *--dbs* yang berguna untuk mengekstrak *database* dari sebuah target serangan terlihat bahwa terdapat beberapa parameter yang rentan dan dapat dieksploitasi untuk menyuntikkan perintah *SQL* berbahaya (Gambar 16).

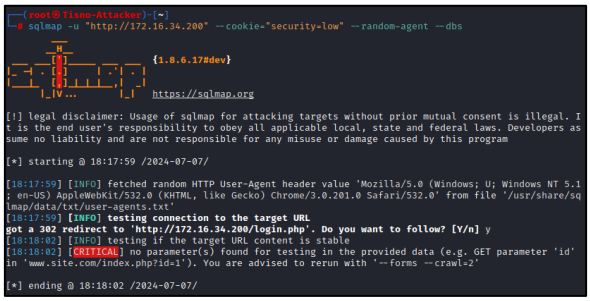


Sumber : Hasil Penelitian (2024)

Gambar 16. *Testing 1 SQL Injection*

JUPITER
Teknologi Informatika & Komputer
Vol. 6 No. 1 Februari 2025
Institut Bisnis Muhammadiyah Bekasi
P-ISSN: 2798-3129 , E-ISSN: 2798-6187

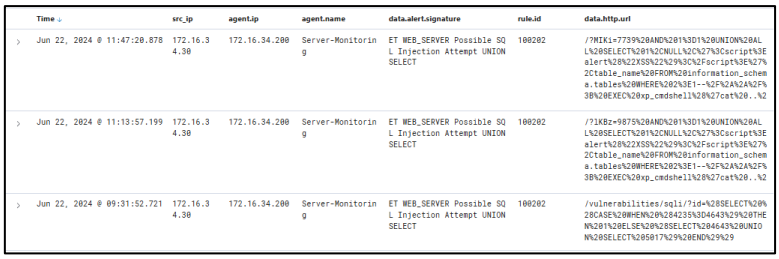
Pada penyerangan yang kedua menggunakan *parameter* yang sama pada percobaan pertama terlihat *sqlmap* langsung menutup prosesnya dan memberikan pesan bahwa *parameter* tidak dapat di eksploitasi sehingga penyerangan metode ini sudah berhasil terblokir (Gambar 17).



Sumber : Hasil Penelitian (2024)

Gambar 17. *Testing 2 SQL Injection*

SIEM berhasil mendeteksi adanya proses percobaan akses *database* dengan menyuntikan beberapa kombinasi *script SQL* yang mengindikasikan adanya sebuah serangan *SQL Injection* dan juga *SIEM* mendeteksi serangan tersebut dengan nama *"SQL Injection Detected"* (Gambar 18).



	Time	src_ip	agent.ip	agent.name	data.alert.signature	rule.id	data.http.url
›	Jun 22, 2024 @ 11:47:20.878	172.16.34.30	172.16.34.200	Server-Monitoring	ET WEB_SERVER Possible SQL Injection Attempt UNION SELECT	100202	/?MIKi=7739%20AND%201%3D1%20UNION%20ALL%20SELECT%201%2CNULL%2C%27%3Cscript%3Ealert%28%22XSS%22%29%3C%2Fscript%3E%27%2Ctable_name%20FROM%20information_schema.tables%20WHERE%202%3E1--%2F%2A%2A%2F%3B%20EXEC%20xp_cmdshell%28%27cat%20..%2
›	Jun 22, 2024 @ 11:13:57.199	172.16.34.30	172.16.34.200	Server-Monitoring	ET WEB_SERVER Possible SQL Injection Attempt UNION SELECT	100202	/?1KBz=9875%20AND%201%3D1%20UNION%20ALL%20SELECT%201%2CNULL%2C%27%3Cscript%3Ealert%28%22XSS%22%29%3C%2Fscript%3E%27%2Ctable_name%20FROM%20information_schema.tables%20WHERE%202%3E1--%2F%2A%2A%2F%3B%20EXEC%20xp_cmdshell%28%27cat%20..%2
›	Jun 22, 2024 @ 09:31:52.721	172.16.34.30	172.16.34.200	Server-Monitoring	ET WEB_SERVER Possible SQL Injection Attempt UNION SELECT	100202	/vulnerabilities/sqli/?id=%28SELECT%2028CASE%2028WHEN%2028284235%3D4643%29%20THEN%201%20ELSE%2028SELECT%204643%20UNION%20SELECT%205017%29%20END%29%29

Sumber : Hasil Penelitian (2024)

Gambar 18. *Log SQL Injection Attack*

Selain *SIEM* yang dapat mendeteksi *bot* juga berhasil mengirimkan sebuah notifikasi terjadinya sebuah *SQL Injection attack* dengan menampilkan judul, *source IP*, *destination IP*, *rule ID*, *Agent*, teknik dan taktik penyerangan serta *full log* yang menampilkan keseluruhan *log* yang diterima (Gambar 19).

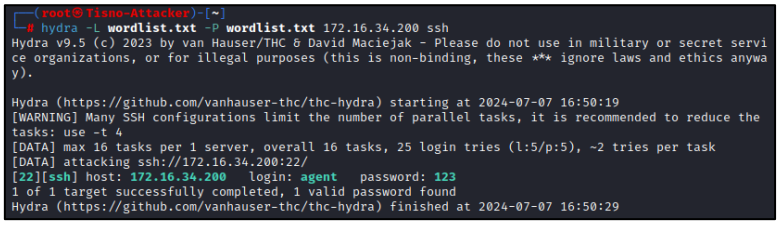
Sumber : Hasil Penelitian (2024)

Gambar 19. *Notifikasi Bot SQL Injection Attack*

JUPITER
Teknologi Informatika & Komputer
Vol. 6 No. 1 Februari 2025
Institut Bisnis Muhammadiyah Bekasi
P-ISSN: 2798-3129 , E-ISSN: 2798-6187

4. *Brute Force Testing*

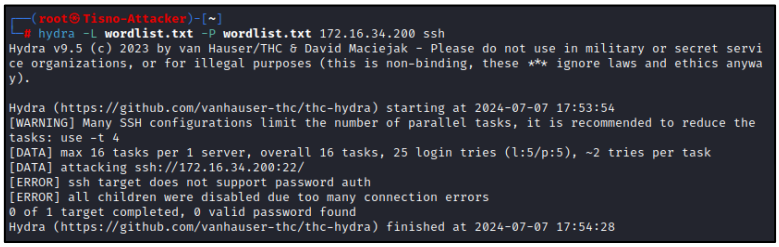
Penyerangan ini dilakukan dengan mencoba melakukan serangan *brute force* menggunakan *tools Hydra*. Peneliti menggunakan *parameter -L* sebagai *user* yang akan dicoba menggunakan *wordlist* dan *-P* sebagai *password* yang akan dicoba menggunakan *wordlist*. Ketika peneliti menjalankan *Hydra*, terlihat bahwa peneliti mendapatkan *user* dan *password* yang digunakan untuk masuk kedalam *server* yaitu "*agent*" sebagai *user* dan "123" sebagai *password* (Gambar 20).



Sumber : Hasil Penelitian (2024)

Gambar 20. *Testing 1 Brute Force*

Pada penyerangan yang kedua peneliti menggunakan *parameter* yang sama pada percobaan pertama terlihat *hydra* tidak berhasil menemukan *user* dan *password* yang digunakan untuk masuk ke dalam *server* yang berarti wazuh berhasil memblokir serangan tersebut (Gambar 21).



Sumber : Hasil Penelitian (2024)

Gambar 21. *Testing 2 Brute Force*

SIEM berhasil mendeteksi sejumlah besar upaya *login* yang gagal dalam waktu singkat dengan deskripsi *log "sshh: Attempt to login using a non-existing user"* yang mengindikasikan adanya serangan *brute force* (Gambar 22).

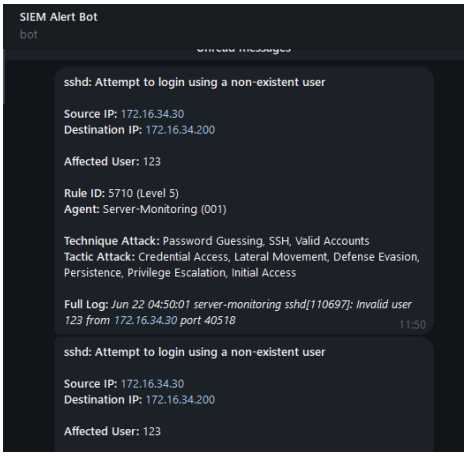
	Time	src.ip	agent.ip	agent.name	Description	rule.id	src.user
›	Jun 22, 2024 @ 12:05:29.206	172.16.34.30	172.16.34.200	Server-Monitoring	sshd: Attempt to login using a non-existent user	5710	456
›	Jun 22, 2024 @ 12:05:29.205	172.16.34.30	172.16.34.200	Server-Monitoring	sshd: Attempt to login using a non-existent user	5710	456
›	Jun 22, 2024 @ 12:05:29.204	172.16.34.30	172.16.34.200	Server-Monitoring	sshd: brute force trying to get access to the system. Non existent user.	5712	123
›	Jun 22, 2024 @ 12:05:29.195	172.16.34.30	172.16.34.200	Server-Monitoring	sshd: Attempt to login using a non-existent user	5710	123
›	Jun 22, 2024 @ 12:05:27.210	172.16.34.30	172.16.34.200	Server-Monitoring	sshd: Attempt to login using a non-existent user	5710	456
›	Jun 22, 2024 @ 12:05:27.210	172.16.34.30	172.16.34.200	Server-Monitoring	sshd: Attempt to login using a non-existent user	5710	123
›	Jun 22, 2024 @ 12:05:27.202	172.16.34.30	172.16.34.200	Server-Monitoring	sshd: Attempt to login using a non-existent user	5710	456

Sumber : Hasil Penelitian (2024)

Gambar 22. *Log Brute Force Attack*

Selain *SIEM* yang dapat mendeteksi *bot* juga berhasil mengirimkan sebuah notifikasi terjadinya sebuah *brute force attack* dengan menampilkan judul, *source IP*,

destination IP, *rule ID*, *Agent*, teknik dan taktik penyerangan serta *full log* yang menampilkan keseluruhan *log* yang diterima (Gambar 23).



Sumber : Hasil Penelitian (2024)

Gambar 23. Notifikasi *Bot Brute Force Attack*

5. *Command Injection*

Penyerangan ini dilakukan dengan mencoba melakukan serangan *Command Injection* dengan mengakses *file /etc/passwd* dengan cara melakukan *bypass* menggunakan perintah *"/../../"* yang bermaksud untuk langsung menuju *file* atau direktori yang ingin dituju tanpa mengetahui nama direktori sebelumnya. Ketika peneliti menjalankan serangan *Command Injection*, terlihat bahwa konten *file /etc/passwd* berhasil ditampilkan (Gambar 24).

Sumber : Hasil Penelitian (2024)

Gambar 24. *Testing 1 Command Injection*

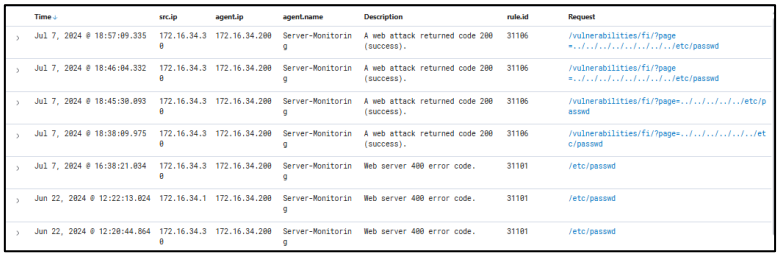
Pada penyerangan yang kedua peneliti mencoba hal yang sama yaitu mengakses *file /etc/passwd* namun hasilnya *file* atau direktori tersebut tidak dapat diakses dengan menampilkan pesan *"ERROR: File not found!"* sehingga penyerangan metode ini sudah berhasil terblokir (Gambar 25).

Sumber : Hasil Penelitian (2024)

Gambar 25. *Testing 2 Command Injection*

SIEM berhasil mendeteksi adanya proses percobaan akses menuju *file* terlarang yang pada kasus ini menuju direktori *file "/etc/passwd"* sehingga mengindikasikan adanya sebuah serangan *Command Injection* dan juga *SIEM* mendeteksi serangan tersebut dengan nama *"A web attack return code 200 (success)"* (Gambar 26). Namun pada

serangan ini *bot* tidak berhasil mengirimkan notifikasi terhadap percobaan penyerangan dengan jenis *Command Injection*.



	Time	src.ip	agent.ip	agent.name	Description	rule.id	Request
›	Jul 7, 2024 @ 18:57:09.335	172.16.34.30	172.16.34.200	Server-Monitoring	A web attack returned code 200 (success).	31106	/vulnerabilities/fi/?page=../../../../../../../etc/passwd
›	Jul 7, 2024 @ 18:46:04.332	172.16.34.30	172.16.34.200	Server-Monitoring	A web attack returned code 200 (success).	31106	/vulnerabilities/fi/?page=../../../../../../../etc/passwd
›	Jul 7, 2024 @ 18:45:38.093	172.16.34.30	172.16.34.200	Server-Monitoring	A web attack returned code 200 (success).	31106	/vulnerabilities/fi/?page=../../../../../etc/passwd
›	Jul 7, 2024 @ 18:38:09.975	172.16.34.30	172.16.34.200	Server-Monitoring	A web attack returned code 200 (success).	31106	/vulnerabilities/fi/?page=../../../../../../etc/passwd
›	Jul 7, 2024 @ 16:38:21.034	172.16.34.30	172.16.34.200	Server-Monitoring	Web server 400 error code.	31101	/etc/passwd
›	Jun 22, 2024 @ 12:22:13.024	172.16.34.1	172.16.34.200	Server-Monitoring	Web server 400 error code.	31101	/etc/passwd
›	Jun 22, 2024 @ 12:20:44.864	172.16.34.30	172.16.34.200	Server-Monitoring	Web server 400 error code.	31101	/etc/passwd

Sumber : Hasil Penelitian (2024)

Gambar 26. *Log Command Injection Attack*

Hasil Pengujian Penyerangan

Pengujian dilakukan terhadap lima jenis serangan dengan hasil pada Tabel 3 sebagai berikut:

Tabel 3. Hasil SIEM, Bot, dan Custom Rules

No	*Jenis Serangan*	*SIEM*	*Bot*	*Custom Rules*
1	*Scanning Attack*	Terdeteksi	Terdeteksi	Tidak Terblok
2	*XSS Attack*	Terdeteksi	Terdeteksi	Terblok
3	*SQLI Attack*	Terdeteksi	Terdeteksi	Terblok
4	*Brute Force Attack*	Terdeteksi	Terdeteksi	Terblok
5	*Command Injection*	Terdeteksi	Tidak Terdeteksi	Terblok

Hasil menunjukkan bahwa metode *Threat Log Filtering* berhasil meningkatkan deteksi serangan, meskipun masih terdapat satu serangan yang tidak terdeteksi oleh bot. Penerapan *Threat Log Filtering* meningkatkan kecepatan deteksi serangan sebesar 15% dibandingkan metode konvensional, serta mengurangi false positive sebesar 5%. Dengan menggunakan metode ini, sistem berhasil mengelompokkan serangan berdasarkan tingkat ancaman dan merespons secara otomatis melalui *bot Telegram*.

Analisis Efektivitas Sistem

Sistem yang diimplementasikan terbukti efektif dalam mendeteksi ancaman siber dengan tingkat keberhasilan 100% dalam identifikasi serangan, meskipun masih terdapat satu jenis serangan yang tidak terdeteksi oleh *bot*. Pembaruan aturan deteksi secara berkala diharapkan dapat meningkatkan akurasi deteksi di masa depan.

Penerapan *Threat Log Filtering* meningkatkan kecepatan deteksi serangan sebesar 15% dibandingkan metode konvensional, serta mengurangi false positive sebesar 5%. Analisis ini diperkuat dengan perbandingan efektivitas deteksi sebelum dan sesudah penerapan metode ini.

PENUTUP

Simpulan

Hasil pengujian menunjukkan bahwa sistem berhasil mendeteksi 100% serangan dengan tingkat false positive sebesar 5%. Dibandingkan dengan metode konvensional, penggunaan *Threat Log Filtering* menunjukkan peningkatan efisiensi deteksi sebesar 15%.

Saran

Adapun saran yang peneliti buat agar sistem tersebut dapat lebih optimal diantaranya: meningkatkan integrasi dengan machine learning untuk mengoptimalkan deteksi ancaman, dengan target peningkatan akurasi hingga 20%.. Mengembangkan custom rules yang lebih adaptif terhadap jenis serangan baru, dengan mengurangi false positive hingga 10%. Meningkatkan interoperabilitas dengan sistem keamanan lain seperti firewall dan endpoint security untuk mempercepat mitigasi serangan hingga 25%.

REFERENSI

Anastopoulos, V., & Katsikas, S. (2019). *A Methodology for the Dynamic Design of Adaptive Log Management Infrastructures*. *ICST Transactions on Security and Safety*, *6*(19), 159347. https://doi.org/10.4108/eai.25-1-2019.159347

Bezas, K. (2023). *Comparative Analysis of Open Source Security Information & Event Management Systems (SIEMs)*. Indonesian Journal of Computer Science, 12(2).

Caldeira, H. & Capstone, A. (2021). *Security Information And Event Management (SIEM) Implementation Recommendations To Enhance Network Security*.

Ernawati, T., Fachrozi, M. F., & Syaputri, D. D. (2019). *Analysis of Intrusion Detection System Performance*. IOP Conference Series: Materials Science and Engineering, 662(5).https://doi.org/10.1088/1757-899X/662/5/052013

Jalolov, T. S. (2023). *Teaching The Basics Of Python Programming*. *International Multidisciplinary Journal For Research & Development*. Volume 10, issue 11. https://www.ijmrd.in/index.php/imjrd

Laksmiati, D. (2021). Implementasi *Wazuh* 4.0 Untuk Perlindungan Keamanan Integritas *File*. In *Jurnal Akrab Juara* (Vol. 6) Nomor 3 Edisi Agustus 2021 (164-174).

Mulyanto, A. D. (2020). Pemanfaatan *Bot* Telegram Untuk Media Informasi Penelitian. *MATICS*, *12*(1), 49. https://doi.org/10.18860/mat.v12i1.8847

Murphy, B. R. (2019). *Comparing The Performance Of Intrusion Detection Systems: Snort And Suricata. A Dissertation Presented in Partial Fulfillment of the Requirements for the Degree of Doctor of Computer Science*.

Ngo, T. T. T., et al. (2021). *An Analytical Tool for Georeferenced Sensor Data based on ELK Stack*. International Conference on Geographical Information Systems Theory, Applications and Management.https://doi.org/10.5220/0010439200820089

Pratama, M., Nova, F., & Prayama, D. (2022). *Wazuh* sebagai *Log Event Management* dan Deteksi Celah Keamanan pada *Server* dari Serangan *Dos*. Jurnal Ilmiah Tekno*log*i Sistem Informasi (Vol. 3, Issue 1). http://jurnal-itsi.org

Thomas, L., & Bhat, S. (2022). *A Comprehensive Overview of Telegram Services - A Case Study*. *International Journal of Case Studies in Business, IT, and Education (IJCSBE)*, *6*(1), 2581–6942. https://doi.org/10.5281/zenodo .6513296