

# ANALISIS FILE LOG APACHE SERVER TERHADAP DETEKSI SERANGAN DICTIONARY ATTACK PADA FORM LOGIN APLIKASI WEB

Muhammad Rafi Bayu Saputra<sup>1\*</sup>, Rihartanto<sup>2</sup>, Ansar Rizal<sup>3</sup>

<sup>1,2,3</sup>Politeknik Negeri Samarinda; Jl. Dr. Cipto Mangunkusumo, Samarinda 75131;

## Keywords:

Brute Force, Apache, File Log, Keamanan Web, Burp Suite.

## Correspondent Email:

m.raficool35@gmail.com

**Abstrak.** Keamanan aplikasi web menjadi salah satu aspek krusial dalam infrastruktur jaringan saat ini. Salah satu ancaman yang paling sering dieksploitasi adalah serangan brute force, khususnya menggunakan teknik dictionary attack pada halaman autentikasi. Penelitian ini bertujuan untuk menganalisis efektivitas pemantauan file log pada Apache Web Server dalam mendeteksi anomali serangan tersebut. Pengujian dilakukan secara lokal menggunakan Damn Vulnerable Web Application (DVWA) sebagai target dan Burp Suite sebagai alat simulasi serangan. Hasil penelitian menunjukkan bahwa serangan dictionary attack meninggalkan jejak digital yang masif pada access.log, berupa lonjakan permintaan metode POST dalam rentang waktu sepersekian detik. Kesimpulan dari penelitian ini menegaskan bahwa implementasi log monitoring merupakan langkah fundamental yang sangat efektif untuk deteksi dini dan investigasi forensik terhadap ancaman keamanan server.



Copyright © 2026 The Author(s), Published by Jurnal Informatika dan Teknik Elektro Terapan. This is an open-access article distributed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

**Abstract.** Web application security has become one of the most critical aspects of modern network infrastructure. One of the most frequently exploited threats is a brute-force attack, particularly through the use of dictionary attack techniques targeting authentication pages. This study aims to analyze the effectiveness of monitoring Apache Web Server log files in detecting anomalies caused by such attacks. The experiments were conducted in a local environment using Damn Vulnerable Web Application (DVWA) as the target application and Burp Suite as the attack simulation tool. The results indicate that dictionary attacks leave significant digital traces in the Apache access.log file, characterized by a sharp surge in HTTP POST requests within fractions of a second. The findings confirm that log monitoring is a fundamental and highly effective approach for early detection and forensic investigation of web server security threats.

## 1. PENDAHULUAN

Dalam era digital yang terinterkoneksi saat ini, aplikasi web sering kali menjadi titik masuk utama (*entry point*) sekaligus target eksploitasi oleh pihak yang tidak bertanggung jawab. Kerentanan pada sistem autentikasi masih menjadi celah keamanan yang masif. Salah satu

ancaman keamanan konvensional yang persisten dan memiliki tingkat keberhasilan tinggi adalah serangan *Brute Force* dan *Dictionary Attack*. Serangan ini mengeksploitasi kerentanan pada form autentikasi dengan cara menginjeksi ribuan kombinasi nama pengguna dan kata sandi

secara berulang, masif, dan terotomatisasi. Minimnya kesadaran administrator jaringan terhadap pemantauan lalu lintas data secara *real-time* sering kali menyebabkan serangan intrusi ini baru terdeteksi setelah sistem berhasil diretas dan data dieksploitasi.

Penelitian terkait deteksi intrusi pada *web server* telah banyak dilakukan, namun sebagian besar berfokus pada perancangan sistem deteksi pihak ketiga seperti *Intrusion Detection System* (IDS) atau *Intrusion Prevention System* (IPS) yang kompleks dan memerlukan alokasi sumber daya komputasi tambahan. Bagi administrator *server* skala menengah ke bawah, implementasi sistem keamanan pihak ketiga sering kali terkendala oleh biaya, lisensi, dan spesifikasi perangkat keras. Oleh karena itu, terdapat kesenjangan (*gap*) dalam mengoptimalkan fitur keamanan proaktif bawaan yang sebenarnya sudah tersedia pada arsitektur *web server*, yaitu sistem pencatatan aktivitas (*logging*).

Penelitian ini bertujuan untuk menyimulasikan serangan *dictionary attack* pada lingkungan terkendali secara lokal, dan secara spesifik menganalisis karakteristik jejak forensik serangan tersebut di dalam *file access.log* pada Apache Web Server. Melalui analisis pembedahan parameter log secara mendalam, penelitian ini diharapkan dapat memberikan pembuktian empiris bahwa pemantauan log (*log monitoring*) merupakan langkah mitigasi tingkat pertama yang esensial, efisien secara komputasi, dan efektif dalam mengidentifikasi anomali jaringan sebelum eksploitasi lebih lanjut terjadi.

## 2. TINJAUAN PUSTAKA

Keamanan aplikasi web merupakan aspek krusial dalam menjaga integritas, ketersediaan, dan kerahasiaan data pada *server*. Menurut standar rilis *Open Web Application Security Project* (OWASP), serangan terhadap sistem autentikasi dan kegagalan manajemen sesi (*Identification and Authentication Failures*) masih menduduki posisi atas sebagai celah kerentanan dengan tingkat risiko tertinggi [1], [2]. *Dictionary attack* merupakan variasi taktis dari *brute force* yang bekerja secara otomatis untuk menebak kredensial pengguna. Alih-alih mencoba kombinasi huruf acak, metode ini memanfaatkan basis data kata sandi (*wordlist*)

yang berisi kumpulan kata sandi yang paling umum digunakan, mudah ditebak, atau pernah bocor di internet melalui insiden *data breach* [3], [4]. Otomatisasi menggunakan perangkat lunak penetrasi memungkinkan penyerang mengirimkan ratusan hingga ribuan permintaan (*request*) HTTP ke *server* hanya dalam hitungan detik [5].

Untuk memitigasi risiko tersebut, pemantauan aktivitas *server* secara proaktif atau *Log Monitoring* menjadi garis pertahanan fundamental yang sering terabaikan [6]. Apache HTTP Server, sebagai salah satu *web server open-source* paling populer di dunia, memiliki mekanisme *logging* bawaan (biasanya menggunakan format *Common Log Format* atau *Combined Log Format*) yang mendokumentasikan setiap transaksi klien-server pada *file* terpusat bernama *access.log* [7]. *File* teks ini merekam metrik dan parameter penting seperti alamat IP klien yang melakukan *request*, cap waktu (*timestamp*) presisi, metode permintaan HTTP (GET, POST, PUT, dll), *Uniform Resource Identifier* (URI) yang dituju, status kode respons dari *server*, hingga agen pengguna (*User-Agent*) [8], [9].

Dalam implementasi keamanan tingkat lanjut, analisis mendalam dan berkala pada *file log* telah terbukti sebagai metode investigasi forensik digital yang andal untuk mendeteksi anomali *traffic* jaringan. Teknik pemindaian log ini sering kali memanfaatkan Ekspresi Reguler (*Regular Expression / Regex*) untuk memfilter lalu lintas data siber mencurigakan tanpa membebani kinerja RAM atau prosesor *server* [10], [11]. Beberapa penelitian terdahulu yang dipublikasikan pada Jurnal Informatika dan Teknik Elektro Terapan (JITET) mengonfirmasi bahwa penerapan sistem analitis terhadap data masif sangat krusial untuk mencegah eksploitasi dan menghasilkan keputusan keamanan yang akurat [14], [15]. Berlandaskan literatur tersebut, penelitian ini difokuskan pada pengungkapan karakteristik jejak forensik dari serangan *dictionary attack* berdasarkan parameter log standar Apache.

## 3. METODE PENELITIAN

Penelitian ini menggunakan pendekatan eksperimental terstruktur dengan menyimulasikan skenario serangan siber pada lingkungan *localhost* (lingkungan tertutup)

untuk mencegah risiko interupsi eksternal dan kebocoran data.

### 3.1 Kebutuhan Perangkat Keras dan Perangkat Lunak

Untuk melaksanakan pengujian dan simulasi secara optimal, diperlukan spesifikasi lingkungan perangkat keras (*hardware*) dan perangkat lunak (*software*) sebagaimana dijabarkan pada Tabel 1 dan Tabel 2 berikut:

Tabel 1. Spesifikasi Perangkat Keras Pengujian

| Komponen Perangkat Keras | Spesifikasi Minimal / Digunakan               | Deskripsi Fungsional                                                                                |
|--------------------------|-----------------------------------------------|-----------------------------------------------------------------------------------------------------|
| Prosesor (CPU)           | Intel Core i3 / AMD Ryzen 3 (Generasi Setara) | Memproses beban kerja otomatisasi request pada Burp Suite dan merespons traffic pada Apache Server. |
| Memori (RAM)             | 4 GB - 8 GB DDR4                              | Mengelola basis data MySQL dan aplikasi Java (Burp Suite) yang berjalan secara paralel.             |
| Penyimpanan (Storage)    | SSD 256 GB                                    | Mempercepat operasi baca/tulis (read/write) I/O saat pencatatan file access.log secara masif.       |

Tabel 2. Spesifikasi Perangkat Lunak Lingkungan Pengujian

| Komponen Perangkat Lunak | Versi / Spesifikasi      | Fungsi Utama dalam Pengujian    |
|--------------------------|--------------------------|---------------------------------|
| Sistem Operasi           | Windows 10 / 11 (64-bit) | Sistem host tunggal untuk mesin |

|                       |                                |                                                                                                        |
|-----------------------|--------------------------------|--------------------------------------------------------------------------------------------------------|
|                       |                                | penyerang dan mesin target (localhost).                                                                |
| Web Server & Database | XAMPP (Apache v2.4, MySQL)     | Penyedia layanan web server lokal dan sistem pencatatan aktivitas (log generator).                     |
| Aplikasi Target       | DVWA (Damn Vulnerable Web App) | Platform web simulasi yang sengaja dirancang memiliki kerentanan pada parameter form login.            |
| Instrumen Penyerangan | Burp Suite Community Edition   | Alat utama berbasis Proxy dan Intruder untuk mencegah paket data dan mengeksekusi otomatisasi payload. |

### 3.2 Skenario Pengujian

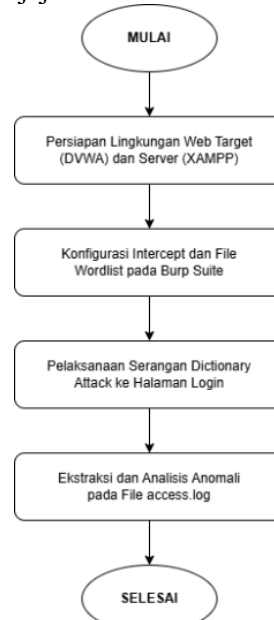
Skenario pengujian dirancang untuk menyimulasikan serangan tebak sandi yang realistis. Serangan tidak dilakukan secara acak (bukan murni *Brute Force* yang mencoba seluruh kombinasi abjad), melainkan menggunakan teknik *Dictionary Attack*. Sebuah file teks kustom disiapkan sebagai kamus (*wordlist*). Kamus ini berisi deretan kata sandi yang dikurasi, yang merepresentasikan kata sandi lemah yang paling umum digunakan oleh pengguna awam (misal: "123456", "admin123", "password", "qwerty"). Keberhasilan skenario ini diukur dari kemampuan penyerang menemukan kata sandi yang valid, serta kemampuan sistem *server* mencatat jejak serangan tersebut tanpa interupsi.

### 3.3 Tahapan Pelaksanaan Penelitian

Metodologi pelaksanaan penelitian dirancang melalui empat tahapan utama yang saling berkesinambungan:

1. **Persiapan dan Konfigurasi Lingkungan Target:** Tahap ini melibatkan instalasi DVWA pada direktori lokal htdocs milik XAMPP. Selanjutnya, dilakukan konfigurasi *database* MySQL dan penyesuaian level keamanan DVWA pada tingkat "Low" agar *server* dapat menerima *request* tanpa hambatan enkripsi pengaman (seperti CAPTCHA, filter IP, atau CSRF token), guna menyimulasikan sistem web konvensional yang tidak memiliki pengamanan berlapis.
2. **Konfigurasi Instrumen Penyerangan (Intercept):** Pengaturan vektor serangan dilakukan menggunakan fitur *Proxy* pada Burp Suite. Burp Suite dikonfigurasi untuk berdiri sebagai "jembatan penengah" (*Man-in-the-Middle*) yang mendengarkan (*listening*) secara lokal pada *port* 8080. Fitur ini berfungsi untuk menahan (*intercept*) permintaan HTTP POST yang dikirimkan dari *browser* klien sebelum paket data tersebut sampai ke Apache Server, sehingga peretas dapat memodifikasi *header* dan isi paket tersebut secara leluasa.
3. **Pelaksanaan Serangan (Dictionary Attack):** Simulasi eksploitasi masif dijalankan melalui modul *Intruder* tipe *Sniper* pada Burp Suite. Modul ini bertugas mendistribusikan *payload* dari *file wordlist* yang telah disiapkan sebelumnya secara otomatis. Sistem akan menargetkan variabel parameter *password*, menggantinya secara iteratif, dan menembakkannya secara beruntun ke halaman autentikasi DVWA (login.php).
4. **Ekstraksi dan Analisis Forensik Log:** Tahap investigasi difokuskan pada pembedahan *file access.log* yang direkam langsung oleh subsistem *logs* Apache. Data mentah diekstraksi pasca-serangan untuk memetakan

anomali *traffic*, menyeleksi pola frekuensi metode HTTP, dan merumuskan parameter forensik pasti dari jejak intrusi.



Gambar 1. Diagram Alir / Flowchart

#### 4. HASIL DAN PEMBAHASAN

Bagian ini memaparkan secara sistematis hasil dari eksekusi metodologi yang telah dirancang, mulai dari intersepsi data hingga analisis forensik pada pencatatan *server*.

##### 4.1 Persiapan dan Intersepsi Paket Data

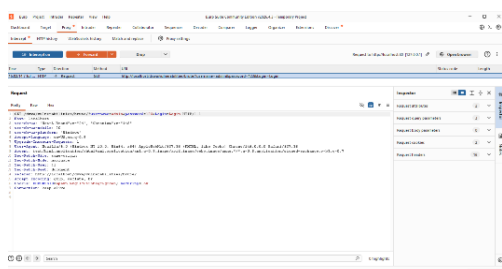
Pengujian diawali dengan validasi lingkungan target untuk memastikan fungsionalitas inti Apache Server dan layanan basis data MySQL berjalan normal. Halaman autentikasi web DVWA diakses melalui *browser* klien pada alamat *localhost/dvwa* guna memperoleh parameter HTTP *request* awal yang absah (Gambar 2).



Gambar 2. Halaman Login Web Target (DVWA)

Untuk menginisiasi *dictionary attack*, jalur komunikasi antara klien dan *server* diinterupsi secara sengaja menggunakan fasilitas *Proxy*

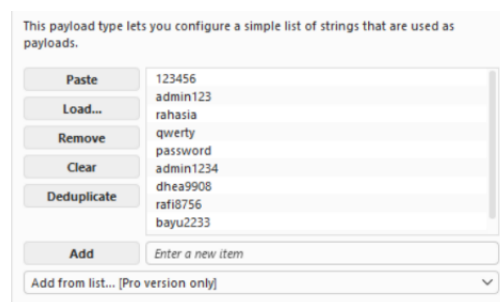
*Intercept* dari Burp Suite (Gambar 3). Melalui mekanisme manipulasi ini, penguji mengirimkan *request login* tiruan (menggunakan *username* 'admin' dan sembarang karakter sebagai *password*). Burp Suite berhasil menahan (*intercept*) paket data HTTP tersebut sesaat sebelum diproses oleh fungsi PHP pada *server* target. Intersepsi ini bersifat krusial dalam metodologi serangan karena memberikan akses penuh bagi penyerang untuk membaca elemen *header* spesifik (seperti *Cookie ID* dan *User-Agent*) serta membuka ruang untuk memanipulasi *body request* yang memuat parameter kredensial.



Gambar 3. Penahanan paket data (*Intercept*)

#### 4.2 Injeksi Payload dan Eksekusi Serangan

Paket data yang berhasil ditahan pada tahap sebelumnya diteruskan ke modul *Intruder*. Pada fase persiapan injeksi ini, penyerang menentukan titik serangan (*payload position*) secara visual dengan menyorot (*highlight*) variabel kata sandi yang dikirimkan. Variabel dinamis tersebut kemudian diintegrasikan secara berantai dengan *file wordlist* yang memuat serangkaian kombinasi kata sandi terkurasi melalui tab konfigurasi *Payloads* (Gambar 4). Mekanisme otomatisasi ini menginstruksikan Burp Suite untuk mengeksekusi mode serangan *Sniper*, di mana sistem secara presisi hanya akan mengganti nilai variabel *password* secara sekuensial pada setiap iterasi pengiriman *request*, tanpa mengubah atau merusak struktur *header* HTTP standar lainnya agar *server* tidak mencurigai *request* tersebut sebagai *malformed packet*.



Gambar 4. Konfigurasi kamus kata sandi

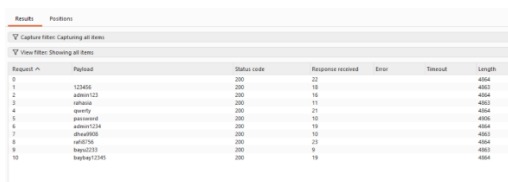
Proses eksekusi serangan *bot* otomatis ini direkam dan disajikan pada Gambar 5. Perangkat lunak Burp Suite tidak sekadar mentransmisikan data, melainkan juga secara aktif merekam dan menganalisis metrik respons balik (*Response Metric*) dari *server* Apache untuk setiap *payload* tunggal yang ditembakkan. Berdasarkan analisis matriks hasil serangan, ditemukan adanya divergensi (*anomaly gap*) yang mencolok pada metrik ukuran respons (*Length*), sebagaimana diuraikan secara numerik pada Tabel 3 berikut.

Tabel 3. Skenario Injeksi Payload dan Metrik Respons Server

| Iterasi ke- | Injeksi Payload ( <i>Password</i> ) | Status HTTP | Ukuran Respons ( <i>Length</i> ) | Indikasi Keberhasilan      |
|-------------|-------------------------------------|-------------|----------------------------------|----------------------------|
| 1           | 123456                              | 200 OK      | 4863 bytes                       | Gagal (Akses Ditolak)      |
| 2           | admin123                            | 200 OK      | 4864 bytes                       | Gagal (Akses Ditolak)      |
| 3           | rahasia                             | 200 OK      | 4863 bytes                       | Gagal (Akses Ditolak)      |
| 4           | qwerty                              | 200 OK      | 4864 bytes                       | Gagal (Akses Ditolak)      |
| 5           | password                            | 200 OK      | 4906 bytes                       | Berhasil (Akses Diberikan) |
| 6           | admin1234                           | 200 OK      | 4864 bytes                       | Gagal (Akses Ditolak)      |
| 7           | dhea9908                            | 200 OK      | 4863 bytes                       | Gagal (Akses Ditolak)      |
| 8           | rafi8756                            | 200 OK      | 4864 bytes                       | Gagal (Akses Ditolak)      |
| 9           | bayu2233                            | 200 OK      | 4863 bytes                       | Gagal (Akses Ditolak)      |
| 10          | baybay12345                         | 200 OK      | 4864                             | Gagal (Akses Ditolak)      |

Analisis mendalam pada Tabel 3 mengungkapkan sebuah kelemahan desain keamanan yang lazim ditemukan pada aplikasi web dasar. Terlihat dengan jelas bahwa sistem *server* tidak memberikan status HTTP "401 Unauthorized" atau "403 Forbidden" ketika input kata sandi terdeteksi salah. Aplikasi justru mengembalikan status "200 OK" pada seluruh percobaan, yang mengindikasikan bahwa halaman dirender dengan sukses. Oleh karena itu, metrik *Length* menjadi parameter teknis sekunder, namun menjadi satu-satunya indikator yang membedakan kegagalan dan keberhasilan eksploitasi dalam simulasi peretas.

Ketika *payload* "password" (yang merupakan entri kata sandi sah untuk akun *admin* pada DVWA) terkirim, ukuran respons melesat secara spesifik menjadi 4906 *bytes*. Perubahan ini terjadi akibat adanya pengalihan *routing* struktural dari sistem, yang memuat komponen HTML tambahan dari halaman dasbor administrator, alih-alih merender ulang form *login*. Anomali ukuran *Length* inilah yang dikonversi dan disimpulkan oleh peretas sebagai indikator absolut keberhasilan penetrasi sistem, tanpa penyerang harus meninjau halaman web secara visual.

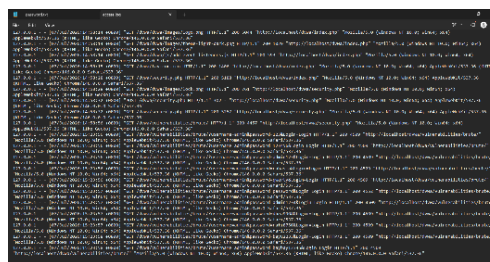


| No. | Request   | Response | Status code | Response received | Error | Timestamp | Length |
|-----|-----------|----------|-------------|-------------------|-------|-----------|--------|
| 0   |           |          | 200         | 10                |       | 0.000000  | 4906   |
| 1   | 123456    |          | 200         | 10                |       | 0.000000  | 4906   |
| 2   | admin123  |          | 200         | 10                |       | 0.000000  | 4906   |
| 3   | admin     |          | 200         | 10                |       | 0.000000  | 4906   |
| 4   | admin     |          | 200         | 10                |       | 0.000000  | 4906   |
| 5   | password  |          | 200         | 10                |       | 0.000000  | 4906   |
| 6   | admin1234 |          | 200         | 10                |       | 0.000000  | 4906   |
| 7   | admin123  |          | 200         | 10                |       | 0.000000  | 4906   |
| 8   | admin123  |          | 200         | 10                |       | 0.000000  | 4906   |
| 9   | admin123  |          | 200         | 10                |       | 0.000000  | 4906   |

Gambar 5. Proses simulasi serangan Brute Force menggunakan fitur Intruder

#### 4.3 Analisis Jejak Forensik pada File *access.log*

Aktivitas masif dari eksploitasi otomatis yang terjadi pada tahap 4.2 ternyata meninggalkan jejak forensik digital yang sangat transparan, komprehensif, dan tidak dapat dihapus oleh penyerang dari luar sistem. Pencatatan ini terjadi berkat arsitektur pencatatan bawaan Apache. Berdasarkan ekstraksi murni *file access.log* pada direktori log *server* lokal (Gambar 6), ditemukan adanya insiden kritis berupa kepadatan volume permintaan HTTP (*request flooding*) yang sangat anomali.



Gambar 6. Rekaman jejak serangan pada *access.log*

Untuk memahami karakteristik serangan dari sudut pandang pertahanan siber, berikut adalah pembedahan sintaksis *log parsing* dari salah satu baris log yang terekam secara persis pada *access.log* saat simulasi serangan berada pada puncaknya:

```
127.0.0.1 - - [07/Jul/2026:15:23:56 +0800] "GET /dvwa/vulnerabilities/brute/?username=admin&password=123456&Login=Login HTTP/1.1" 200 4509
```

Berdasarkan format baku Apache, baris rekaman di atas memberikan intelegensi data yang sangat berharga bagi administrator:

1. **Alamat IP (127.0.0.1):** Menunjukkan identitas jaringan asal penyerang. Dalam simulasi ini tercatat sebagai *localhost*, namun pada skenario nyata, ini adalah alamat IP publik peretas yang menjadi target pemblokiran utama.
2. **Timestamp ([07/Jul/2026:15:23:56 +0800]):** Mencatat waktu presisi terjadinya aktivitas. Pada rekaman log keseluruhan, terlihat waktu ini berulang puluhan kali di detik yang sama (15:23:56).
3. **Metode dan URI (GET /dvwa/.../?username=admin&password=123456...):** Mencatat bahwa penyerang mencoba mengirimkan parameter secara paksa. *Password* yang terekam akan berubah-ubah pada baris log selanjutnya secara konsisten sesuai isi *wordlist*.
4. **Status dan Panjang Data (200 4509):** Menandakan respons *server* (200 OK) beserta ukuran data yang ditransmisikan kembali ke klien.

#### 4.4 Perbandingan dan Tindakan Mitigasi

Tabel 4 berfungsi untuk merangkum dan membandingkan karakteristik perilaku akses, sebagai tolok ukur perbedaan antara aktivitas log dalam kondisi wajar (organik) dan log yang mengalami interupsi *dictionary attack*.

Tabel 4. Perbandingan Karakteristik Log Normal dan Log Serangan (Dictionary Attack)

| Parameter Analisis Forensik | Interaksi Normal (User Authentication)              | Indikasi Intrusi (Dictionary Attack)                      |
|-----------------------------|-----------------------------------------------------|-----------------------------------------------------------|
| Frekuensi Request Login     | 1 hingga 3 kali percobaan berturut-turut            | Puluhan hingga ratusan request berkelanjutan              |
| Rentang Waktu (Timestamp)   | Selaras dengan rasio kewajaran manusia (jeda menit) | Kecepatan milidetik (kurang dari 1-2 detik secara masif)  |
| Karakteristik URI / Payload | Parameter statis, sesuai daya ingat (input manual)  | Parameter kredensial diinjeksi dan berubah secara mekanis |
| Pola Ukuran Respons Server  | Statis pada satu ukuran respons normal              | Bersifat fluktuatif, menonjol hanya pada payload yang sah |
| Sifat Entitas Akses         | Interaksi organik oleh Manusia                      | Didominasi eksekusi menggunakan Bot atau Skrip Otomatis   |

Berdasarkan analisis empiris pada Tabel 4, terlihat pembuktian forensik yang sangat kontras, khususnya berpusat pada korelasi antara parameter *timestamp* dengan frekuensi *request*. Kecepatan sistem operasi dalam memproses dan menghasilkan puluhan baris rekaman log autentikasi hanya dalam blink of an eye.

waktu sekian detik merupakan bukti absolut adanya eksploitasi terprogram. Interaksi input data dan transit *network* secepat itu secara fisika komputasi mustahil dilakukan secara manual melalui interaksi antarmuka pengguna (*keyboard*) biasa.

#### 4.5 Rekomendasi Tindakan Defensif Berbasis Log Monitoring

Mengacu pada pembedaan *log* yang telah diverifikasi pada sub-bab sebelumnya, administrator sistem jaringan dapat memberdayakan data dari *log monitoring* ini sebagai pilar utama implementasi mitigasi keamanan aktif. Karena log disimpan dalam format *plain text* yang seragam, administrator direkomendasikan untuk menerapkan perangkat lunak sistem proteksi pasif-agresif (seperti *Fail2Ban* di sistem operasi Linux).

*Fail2Ban* bekerja secara harafiah dengan memantau pergerakan data di dalam *access.log* secara *real-time*. Dengan menggunakan fungsi pemfilteran Ekspresi Reguler (*Regex*), sistem dapat diatur sedemikian rupa untuk mengenali pola: "Jika terdapat IP (misalnya 192.168.1.15) yang mengirimkan parameter POST/GET ke URI *login.php* dengan respons kegagalan berturut-turut melebihi batas (*threshold*) 5 kali dalam rentang waktu kurang dari 10 detik", maka sistem mitigasi akan langsung memerintahkan sub-sistem dinding api (*iptables* / *Firewall*) untuk menolak (*DROP*) semua koneksi paket TCP/IP dari alamat IP penyerang tersebut. Pola pertahanan kolaboratif antara Apache Log dan sistem penangkal otomatis ini jauh lebih hemat biaya komputasi dibandingkan mengalokasikan RAM tinggi untuk *Intrusion Prevention System* berbasis kecerdasan buatan.

## 5. KESIMPULAN

Berdasarkan hasil rancangan simulasi pengujian dan analisis mendalam terhadap *file log*, dapat disimpulkan beberapa poin strategis sebagai berikut:

- Serangan *dictionary attack* dapat diidentifikasi, dipetakan, dan diverifikasi secara presisi murni melalui ekstraksi sintaksis *file access.log* pada Apache Web Server. Anomali tersebut ditandai dengan lonjakan abnormal pada frekuensi *request*

otentikasi dalam rentang waktu (milidetik) yang tidak logis bagi interaksi manusia.

- b. Integrasi variabel *timestamp*, metode pengiriman data, struktur URI yang berubah-ubah secara iteratif, serta alamat IP penyerang yang terekam utuh secara *default* di dalam *log* dapat difungsikan sebagai instrumen forensik digital utama bagi administrator jaringan.
- c. Praktik pemantauan log (*log monitoring*) secara proaktif terbukti merupakan lapisan keamanan preventif tingkat pertama yang sangat efektif, terjangkau secara arsitektur, dan tidak membebani komputasi *server*, tanpa mengharuskan infrastruktur bergantung sepenuhnya pada sistem pencegahan intrusi (*Intrusion Prevention System*) pihak ketiga.

#### UCAPAN TERIMA KASIH

Penulis mengucapkan puji syukur dan terima kasih yang sebesar-besarnya kepada **Bapak Rihartanto, ST., M.Si** dan **Bapak Ansar Rizal, S.Kom., MT** selaku dosen pembimbing. Segala dedikasi, arahan, evaluasi teknis, serta bimbingan yang telah diberikan sejak penyusunan proposal hingga selesainya penelitian dan penyusunan jurnal ini sangatlah berharga bagi penulis. Penulis juga menyampaikan apresiasi kepada seluruh staf pengajar dan entitas akademik di Program Studi Teknik Komputer Politeknik Negeri Samarinda yang telah memfasilitasi kelancaran penelitian ini.

#### DAFTAR PUSTAKA

- [1] OWASP Foundation, "OWASP Top 10: 2021- Identification and Authentication Failures," *Open Web Application Security Project*, 2021.
- [2] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network Anomaly Detection: Methods, Systems and Tools," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 303-336, 2014.
- [3] S. A. Aljawarneh, M. B. Yassein, and M. A. Talafha, "A multithreaded brute force attack on web applications," in *Proc. International Conference on Engineering & MIS (ICEMIS)*, Agadir, Morocco, 2016, pp. 1-5.
- [4] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 8th ed. Pearson, 2020.
- [5] J. M. Kizza, *Guide to Computer Network Security*, 5th ed. Springer, 2020.
- [6] M. Roesch, "Snort: Lightweight Intrusion Detection for Networks," in *Proc. 13th USENIX Conference on System Administration (LISA)*, Seattle, WA, 1999, pp. 229-238.
- [7] Apache Software Foundation, "Apache HTTP Server Version 2.4 Documentation - Log Files," 2023. [Online]. Available: <https://httpd.apache.org/docs/2.4/logs.html>.
- [8] R. A. R. Ashfaq, J. Z. Wang, C. Huang, H. Abbas, and Y. L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," *Information Sciences*, vol. 378, pp. 484-497, 2017.
- [9] A. Chuvakin, K. Schmidt, and C. Phillips, *Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management*. Elsevier, 2012.
- [10] S. S. C. Silva, R. M. P. Silva, R. C. G. Pinto, and S. N. Salles, "Botnet detection using log analysis," *Computers & Security*, vol. 31, no. 3, pp. 289-305, 2013.
- [11] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *Journal of Network and Computer Applications*, vol. 84, pp. 25-37, 2017.
- [12] K. Scarfone and P. Mell, "Guide to Intrusion Detection and Prevention Systems (IDPS)," *NIST Special Publication 800-94*, 2007.
- [13] D. Zuech, T. M. Khoshgoftaar, and R. Wald, "Intrusion detection and Big Heterogeneous Data: a Survey," *Journal of Big Data*, vol. 2, no. 3, 2015.
- [14] D. Duei Putri, G. F. Nama, and W. E. Sulistiono, "Analisis Sentimen Kinerja Dewan Perwakilan Rakyat (DPR) Pada Twitter Menggunakan Metode Naive Bayes Classifier," *JITET (Jurnal Informatika dan Teknik Elektro Terapan)*, vol. 10, no. 1, Jan. 2022.
- [15] A. R. Y. P. Dinata and I. Ahmad, "Penerapan Metode Klasifikasi Naive Bayes Terhadap Analisis Sentimen Ulasan Aplikasi MyOrbit," *JITET (Jurnal Informatika dan Teknik Elektro Terapan)*, vol. 11, no. 3, pp. 586-591, 2023.