MENDETEKSI SERANGAN *BACKDOOR SHELL* DAN *BROKEN ACCESS CONTROL* PADA *WEB SERVER* MENGGUNAKAN *SPLUNK*

Program Studi

S1 Sistem Informasi

Oleh:

Sophie Anastasya Putri BR

21410100012

FAKULTAS TEKNOLOGI DAN INFORMATIKA

UNIVERSITAS DINAMIKA

2025

MENDETEKSI SERANGAN *BACKDOOR SHELL* DAN *BROKEN ACCESS CONTROL* PADA *WEB SERVER* MENGGUNAKAN *SPLUNK*

TUGAS AKHIR

Diajukan sebagai salah satu syarat untuk menyelesaikan
Program Sarjana

Oleh:

Nama	**: Sophie Anastasya Putri BR**
NIM	**: 21410100012**
Program Studi	**: S1 Sistem Informasi**

FAKULTAS TEKNOLOGI DAN INFORMATIKA
UNIVERSITAS DINAMIKA
2025

Tugas Akhir

MENDETEKSI SERANGAN *BACKDOOR SHELL* DAN *BROKEN ACCESS CONTROL* PADA *WEB SERVER* MENGGUNAKAN *SPLUNK*

Dipersiapkan dan disusun Oleh

Sophie Anastasya Putri BR

NIM: 21410100012

Telah diperiksa, dibahas dan disetujui oleh Dewan Pembahas

Pada: 22 Agustus 2025

Susunan Dewan Pembahas

Pembimbing

I. Slamet, M.T

NIDN. 0701127503

II. Dr. Anjik Sukmaaji, S.Kom., M.Eng.

NIDN. 0731057301

Pembahas

I. Tutut Wurijanto, M.Kom.

NIDN. 0703056702

Tugas Akhir ini telah diterima sebagai salah satu persyaratan

Untuk memperoleh gelar sarjana

Digitally signed by Julianto
Date: 2025.08.25 17:53:37 +07'00'

Julianto Lemantara, S.Kom., M.Eng.

NIDN. 0722108601

Dekan Fakultas Teknologi dan Informatika

UNIVERSITAS DINAMIKA

PERNYATAAN

PERSETUJUAN PUBLIKASI DAN KEASLIAN KARYA ILMIAH

Sebagai mahasiswa **Universitas Dinamika**, Saya:

Nama	: **Sophie Anastasya Putri BR**
NIM	: **21410100012**
Program Studi	: **S1 Sistem Informasi**
Fakultas	: **Fakultas Teknologi dan Informatika**
Jenis Karya	: **Laporan Tugas Akhir**
Judul Karya	: **MENDETEKSI SERANGAN *BACKDOOR SHELL* DAN *BROKEN ACCESS CONTROL* PADA WEB SERVER MENGGUNAKAN *SPLUNK***

Menyatakan dengan sesungguhnya bahwa:

1. Demi pengembangan Ilmu Pengetahuan, Teknologi dan Seni, Saya menyetujui memberikan kepada **Universitas Dinamika** Hak Bebas Royalti Non-Eksklusif (***Non-Exclusive Royalty Free Right***) atas seluruh isi/sebagian karya ilmiah Saya tersebut diatas untuk disimpan, dialihmediakan, dan dikelola dalam bentuk pangkalan data (*database*) untuk selanjutnya didistribusikan atau dipublikasikan demi kepentingan akademis dengan tetap mencantumkan nama Saya sebagai penulis atau pencipta dan sebagai pemilik Hak Cipta.
2. Karya tersebut diatas adalah hasil karya asli Saya, bukan plagiat baik sebagian maupun keseluruhan. Kutipan, karya, atau pendapat orang lain yang ada dalam karya ilmiah ini semata-mata hanya sebagai rujukan yang dicantumkan dalam Daftar Pustaka Saya.
3. Apabila dikemudian hari ditemukan dan terbukti terdapat tindakan plagiasi pada karya ilmiah ini, maka Saya bersedia untuk menerima pencabutan terhadap gelar kesarjanaan yang telah diberikan kepada Saya.

Demikan surat pernyataan ini Saya buat dengan sebenar-benarnya.

Surabaya, 31 Juli 2025

Sophie Anastasya Putri BR
NIM: 21410100012

"Percayalah kepada Tuhan dengan segenap hatimu, dan janganlah bersandar kepada pengertianmu sendiri. Akuilah Dia dalam segala lakumu, maka Ia akan meluruskan jalanmu."

Sophie Anastasya Putri BR

Laporan Tugas Akhir ini

Saya persembahkan kepada

Keluarga, Dosen Pembimbing, dan

Teman-teman yang saya kasihi

ABSTRAK

Keamanan siber menjadi tantangan utama di era digital, khususnya terhadap serangan seperti *Backdoor Shell* dan *Broken Access Control* yang dapat menyebabkan gangguan operasional hingga kehilangan data. Penelitian ini bertujuan untuk mendeteksi serangan tersebut pada web server dengan memanfaatkan *Splunk* dan fitur *Search Processing Language (SPL)*. Studi kasus dilakukan pada PT Herbal Pharmaceutical, sebuah perusahaan yang bergerak di bidang kesehatan yang mengalami kehilangan data permanen serta adanya gangguan operasional website dan layanan *online* yang sangat merugikan perusahaan. Dengan pendekatan forensik digital proses penelitian meliputi pengumpulan, pemeriksaan, analisis, dan pelaporan data *log* seperti *access log, error log*, dan *syslog*. *SPL* digunakan untuk mengekstraksi pola mencurigakan, seperti akses tidak sah, unggahan *file b374k.php*, atau anomali kode status *HTTP*. Hasil analisis menunjukkan bahwa *SPL* efektif dalam mengidentifikasi pola serangan dengan *Smart* Mode terbukti menjadi yang tercepat dengan durasi 0,823 detik. Mode pencarian *Smart* di *Splunk* terbukti menjadi yang tercepat dalam memproses volume data *log* yang besar, memastikan deteksi ancaman dapat dilakukan secara optimal dan *real-time*. Ditemukan bukti kuat adanya upaya unggahan *file b347k.php* yang berfungsi sebagai *backdoor shell*, serta eksploitasi celah kontrol akses yang memungkinkan penyerang untuk memodifikasi dan menghapus data sensitif. Dengan demikian, penelitian ini berkontribusi pada penguatan sistem pertahanan web melalui deteksi ancaman secara otomatis dan mendukung adaptasi terhadap teknik serangan siber yang terus berkembang.

Kata Kunci: *Backdoor Shell*, *Broken Access Control*, *Splunk*, *Search Processing Language (SPL)*, *Keamanan Siber*, *Log Analysis*, *Forensik Digital*

KATA PENGANTAR

Puji syukur ke hadirat Tuhan Yang Maha Esa yang telah memberikan berkat dan karunia-Nya sehingga penulis dapat menyelesaikan laporan tugas akhir ini dengan judul " MENDETEKSI SERANGAN *BACKDOOR SHELL* DAN *BROKEN ACCESS CONTROL* PADA WEB *SERVER* MENGGUNAKAN *SPLUNK*" ini dengan baik dan lancar. Penyelesaian Tugas Akhir ini sebagai syarat wajib untuk menyelesaikan program sarjana. Tidak terlepas bantuan dari pihak yang telah memberikan masukan, nasihat, saran, kritik, kepada penulis. Oleh karna itu, pada kesempatan ini penulis ingin menyampaikan rasa terimakasih kepada:

1. Papa dan Mama tercinta, serta sanak saudara yang memberikan doa dan dukungan penuh kepada saya
2. Bapak Slamet, M.T selaku Dosen Pembimbing I yang sudah memberikan bimbingan selama proses penyelesaian tugas akhir.
3. Bapak Dr. Anjik Sukmaaji, S.Kom., M.Eng. selaku Dosen Pembimbing II yang sudah memberikan bimbingan selama proses penyelesaian tugas akhir.
4. Bapak Tutut Wurijanto, M.Kom. selaku Dosen Penguji yang telah menguji hasil tugas akhir.
5. Teman-teman perkuliahan di Universitas Dinamika Surabaya yang telah membantu dalam proses penyelesaian tugas akhir.

Penulis menyadari bahwa laporan ini masih jauh dari kata sempurna. Dengan demikian penulis mengharapkan kritik dan saran yang membangun dari pembaca untuk penyempurnaan dalam menyelesaikan laporan. Semoga laporan Tugas Akhir ini dapat bermanfaat untuk penulis sendiri, dan para pembaca.

Surabaya,02 Juli 2025

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

DAFTAR LAMPIRAN

Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Ancaman *siber* merupakan salah satu masalah yang meningkat di dunia *modern* sekaligus era digital ini. (Zhang Y, 2020). Dalam konteks ini, PT. Analis Forensik Digital adalah perusahaan yang bergerak di bidang forensik digital, investigasi forensik perusahaan dan keamanan siber. Bidang spesialisasi forensik digital ini mencakup penyelidikan komersial, kriminal, dan perusahaan, serta masalah litigasi dan penyelesaian sengketa hingga melayani sebagai ahli di pengadilan. Penelitian ini menggunakan studi kasus dari PT Analis Forensik Digital yang melibatkan PT Herbal Pharmaceutical. PT Herbal Pharmaceutical sebuah perusahaan yang bergerak di bidang kesehatan, mengalami serangkaian insiden keamanan yang menunjukkan adanya upaya peretasan terhadap aplikasi web yang mereka gunakan. Penyerangan ini menyebabkan *website* perusahaan mengalami kehilangan data permanen serta adanya gangguan operasional *website* dan layanan *online* yang sangat merugikan perusahaan. Salah satu metode yang digunakan oleh penyerang adalah *backdoor shell*, yang memberikan akses tidak sah ke sistem dan memungkinkan penyerang untuk menjalankan perintah berbahaya. Selain itu, masalah *broken access control* juga menjadi perhatian, di mana penyerang dapat mengeksploitasi kelemahan dalam kontrol akses untuk mendapatkan hak akses yang tidak semestinya.

Analisis *log* merupakan bagian penting dari keamanan siber karena memungkinkan organisasi untuk mendeteksi dan menanggapi ancaman keamanan secara cepat. Namun, metode analisis *log* tradisional sering dilakukan secara manual dan memakan waktu, sehingga sulit untuk mengimbangi volume dan kompleksitas besar data *log*. (Security, 2020).

Entri *log* yang dimanipulasi oleh musuh untuk menyembunyikan aktivitas atau mengeksekusi kode berbahaya merusak integritas dan keamanan sistem dikenal sebagai *log* berbahaya. Penyerang bertujuan untuk mengacaukan tindakan sistem yang sebenarnya. (Haitham Ameen Noman, 2024). Tantangan ini

memerlukan pendekatan inovatif, seperti teknik pembelajaran *machine learning*, untuk meningkatkan akurasi dan efektivitas pendeteksi. (Nashikkar, 2023). *Splunk* sering digunakan untuk menganalisis data mesin secara *real-time*, kemampuan machine learning (ML) *Splunk* memungkinkan mendeteksi pola berbahaya yang mungkin terlewat oleh metode tradisional. Salah satu keunggulan *Splunk* adalah kemampuan untuk memproses dan menganalisis data dengan *Search Processing Language (SPL)*. *SPL* memiliki kemampuan yang kuat untuk membuat *query* pencarian dan deteksi, termasuk identifikasi *log* berbahaya. (Splunk, Search Processing Language (SPL) Overview, 2023).

Salah satu masalah utama dalam mendeteksi *log* adalah kelangkaan data yang timbul dari pengumpulan data terbatas, yang dapat menghambat deteksi yang tepat dan menyebabkan tingkat negatif palsu yang tinggi serta ancaman yang semakin meningkat, penyerang *siber* terus mengembangkan taktik baru, yang membuat deteksi lebih sulit. Dengan bantuan pembelajaran *Machine Learning Splunk* dengan metode Search Processing Language (SPL), mendeteksi *log* dapat mempermudah organisasi untuk menemukan dan menganalisis *anomali* pada trafik jaringan yang mungkin menandakan aktivitas berbahaya. Organisasi dapat mengotomatisasi pencarian, identifikasi, dan mitigasi ancaman *log* berbahaya, yang meningkatkan respons keamanan siber dan menurunkan risiko serangan. (Gupta, 2021). Metode deteksi waktu nyata yang menganalisis atribut *URL* secara *real-time* dapat melawan serangan *phishing* secara efektif. (Jain, 2023).

Adanya bantuan dari *Machine Learning Splunk* dan penggunaan *Search Processing Language (SPL),* penelitian ini bertujuan untuk mengembangkan sistem yang dapat meningkatkan keamanan jaringan perusahaan, personal dan organisasi dengan mendeteksi *log* berbahaya. Penggunaan *Search Processing Language (SPL)* dalam konteks keamanan *siber* adalah bagian penting dari strategi pertahanan *siber* kontemporer karena metode ini tidak hanya membantu dalam mengidentifikasi ancaman tetapi juga memungkinkan penyesuaian dan peningkatan model deteksi seiring dengan perkembangan teknik serangan *siber*.

Solusi ini dapat meningkatkan deteksi ancaman *siber* dan mengurangi kerugian yang disebabkan oleh serangan *log* berbahaya. Meskipun ada kemajuan,

ancaman *siber* yang terus berubah memerlukan penelitian terus menerus dan pengembangan teknik deteksi untuk menjaga infrastruktur digital.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang di atas, maka rumusan masalah dari laporan Tugas Akhir (TA) ini yaitu:

1. Bagaimana cara mendeteksi serangan *Backdoor Shell* dan *Broken Access Control* dalam menerapkan algoritma *Search Processing Language*?
2. Sejauh mana efektivitas penggunaan metode *SPL (Search Processing Language)* dalam proses deteksi serangan *Backdoor Shell* dan *Broken Access Control*?

1.3 Batasan Masalah

Adapun batasan masalah dalam melakukan penelitian ini adalah sebagai berikut:

1. Penelitian ini fokus pada analisis data *access log* yang dihasilkan oleh sistem dan perangkat jaringan, tanpa mencakup jenis data lain.
2. Penelitian ini akan menetapkan kriteria spesifik untuk mengidentifikasi web server diserang, seperti frekuensi akses yang tinggi atau pola perilaku mencurigakan, tanpa membahas semua kemungkinan kriteria yang ada.
3. Fokus penelitian adalah pada penerapan *SPL (Search Processing Language)* untuk mendeteksi web server diserang, sehingga tidak akan membahas aspek lain dari penggunaan *Splunk* yang tidak terkait langsung dengan deteksi *log*.

1.4 Tujuan

Berdasarkan uraian rumusan masalah, maka dapat tujuan yang dicapai adalah:

1. Untuk mendeteksi serangan *Backdoor Shell* dan *Broken Access Control* dalam menerapkan algoritma *Search Processing Language*.
2. Untuk mempercepat proses penggunaan *Search Processing Language (SPL)* dengan cara mengekstrak dan menganalisis fitur-fitur dari *log* yang mencurigakan, sehingga meningkatkan efektivitas deteksi.

1.5 Manfaat

Dari penelitian ini dapat memberikan manfaat sebagai berikut:

1. Untuk meningkatkan keamanan jaringan dengan mendeteksi web server diserang. Organisasi dapat mengurangi potensi serangan *siber* seperti *phishing* dan *malware* serta melindungi data sensitif mereka.
2. *Machine learning* memungkinkan deteksi otomatis yang lebih cepat daripada metode manual, memberi tim keamanan lebih banyak waktu untuk menangani ancaman yang teridentifikasi.
3. Dengan menggunakan analisis berbasis *Search Processing Language (SPL)*, model dapat meningkatkan kepercayaan tim keamanan dalam pengambilan keputusan dengan mengurangi jumlah *false positives* dalam identifikasi *log* berbahaya.
4. Model pembelajaran *machine learning* memiliki kemampuan untuk belajar terus menerus dari data baru, yang memungkinkan mereka untuk menyesuaikan diri dengan teknik serangan terbaru yang mungkin tidak terdeteksi oleh sistem tradisional.

BAB II
LANDASAN TEORI

2.1 Penelitian Terdahulu

Penelitian ini dibangun di atas fondasi studi-studi sebelumnya. Berbagai penelitian yang relevan menjadi rujukan utama untuk mengembangkan pendekatan yang lebih mendalam dan menyeluruh. Mengkaji studi-studi ini membantu penulis memahami metodologi, model, dan temuan yang telah ada, sehingga penelitian ini dapat memberikan kontribusi yang lebih signifikan. Tabel 2.1 merangkum penelitian-penelitian terdahulu yang dijadikan sebagai bahan acuan dalam penelitian ini :

Tabel 2.1 Penelitian Terdahulu

Peneliti	Judul	Hasil	Persamaan & Perbedaan
Wahluf Abidian	Implementasi *Splunk* dalam Membangun *Security Information and Event Management* Berdasarkan *Log Firewall* (studi kasus: Jaringan UII)	Penelitian ini berhasil mengimplementasikan *Splunk* untuk membangun sistem *SIEM* berbasis log *firewall,* dengan hasil utama berupa visualisasi data yang mudah dipahami, deteksi ancaman melalui *alert* terintegrasi, dan analisis klasterisasi untuk identifikasi pola *traffic*. Solusi ini meningkatkan keamanan jaringan UII dan efisiensi kerja administrator.	Persamaan: Menggunakan *Splunk* sebagai alat utama untuk analisis log dan deteksi ancaman keamanan. Membahas topik keamanan informasi, termasuk deteksi serangan dan analisis log untuk meningkatkan keamanan jaringan. Perbedaan : Fokus pada implementasi *SIEM* berbasis *log firewall*. Menggunakan *log firewall* dari *Palo Alto Networks* (*NGFW*).
Muhammad Rijal Kamal	Deteksi Anomali dengan *Security Information and Event Management* (SIEM) *Splunk* pada Jaringan UII	Penelitian ini Membuktikan bahwa *Splunk* efektif untuk deteksi anomali pada *log firewall,* tetapi hasilnya terbatas oleh skala data dan sumber daya *hardware*. Implementasi lebih lanjut dengan dataset lengkap dan optimasi aturan deteksi dapat meningkatkan akurasi dan keandalan sistem.	Persamaan: Menggunakan pendekatan analisis *log* dan visualisasi data untuk mengidentifikasi ancaman. Perbedaan : Berfokus pada deteksi anomali umum di jaringan kampus menggunakan *log firewall*, dengan tantangan utama pada keterbatasan dataset.
Suheni, Al Ahfaz Reza Ramdani, Emikawati &	Penerapan *Splunk* Terhadap Analisis *File Log Anomaly* Keamanan	Penelitian ini terbukti *Splunk* sebagai alat yang efektif untuk deteksi anomali *log* secara *real-time,* terutama dengan	Persamaan: Menggunakan *Splunk* sebagai platform utama untuk analisis *log,* termasuk fitur seperti *Search Processing*

Peneliti	Judul	Hasil	Persamaan & Perbedaan
Yudistira Bagus Pratama		pendekatan *unsupervised learning*. Kemampuan analisisnya yang fleksibel dan skalabel menjadikannya solusi ideal untuk lingkungan IT kompleks dan sistem kustom.	*Language (SPL)* dan *Machine Learning Toolkit.* Perbedaan : Pendekatan tanpa pengawasan (*unsupervised learning*) dengan algoritma *K-means*

2.2 *Backdoor Shell*

Backdoor Shell, sering disebut sebagai *shell web*, adalah skrip berbahaya yang memungkinkan akses jarak jauh yang tidak sah ke server *web*. *Backdoor* ini mengeksploitasi kerentanan dalam aplikasi web, memungkinkan penyerang untuk mengeksekusi perintah, memanipulasi data, dan mempertahankan akses persisten. (Xiaobo Yu, 2023). Web *Shell* dapat dibuat dalam setiap bahasa web yang dapat dibuat skrip, tetapi sebagian besar web *shell* yang sering ditemui adalah skrip *.asp, .aspx, .js, .jsp,* atau *.php*. Web *Shell* dapat sangat sederhana, hanya mengandalkan sejumlah kecil kode untuk dieksekusi. (Shelmire, 2025). Selain itu menggunakan variabel global *PHP* seperti *$_GET, $_POST, $_COOKIE, $_REQUEST*, dan *$_FILES* untuk menerima perintah dari penyerang melalui parameter *URL* atau *form*. Variabel-variabel ini sering digunakan untuk menjalankan perintah sistem secara dinamis sesuai input penyerang serta memanfaatkan fungsi eksekusi sistem seperti *system(), exec(), shell_exec(), passthru(),* dan *popen()* untuk menjalankan perintah *shell* atau *command line* pada server. Kode sering dienkripsi atau di-*obfuscate* agar sulit dibaca dan dideteksi, misalnya menggunakan fungsi *base64_decode(), eval(),* atau *encoding* lain untuk menyembunyikan logika aslinya. Terkadang memiliki fitur *login* sendiri seperti password sederhana di dalam skrip untuk membatasi akses hanya bagi pembuatnya, tetapi tetap tanpa autentikasi sistem resmi. Mengirim *log* atau hasil eksekusi ke penyerang baik melalui *email, HTTP request*, atau metode lain yang tidak terlihat oleh pemilik system. *Backdoor shell* dicirikan oleh kemampuannya menerima dan mengeksekusi perintah secara *remote* melalui variabel global, penggunaan fungsi eksekusi sistem, penyamaran kode, penempatan di lokasi tersembunyi, serta akses tanpa autentikasi resmi. Deteksi biasanya dilakukan dengan analisis kode statis dan pencarian pola khas pada *file* website (Raditya Faisal Waliulua, 2020). *PHP Shell b347k.php* ini adalah

alat yang berguna bagi administrator sistem atau web untuk melakukan pengelolaan jarak jauh tanpa menggunakan *cPanel*, dengan terhubung melalui *SSH*, *FTP*, dan sebagainya. Semua tindakan dilakukan melalui browser web. (Lab, n.d.)

Musuh bisa saja membobol server web dengan *web shell* untuk membuat akses yang terus-menerus ke sistem. *Shell Web* merupakan skrip web yang ditempatkan di server web yang dapat diakses secara terbuka untuk memungkinkan musuh mengakses server web sebagai pintu gerbang ke dalam jaringan. Sebuah *shell web* dapat menyediakan sekumpulan fungsi untuk dieksekusi atau antarmuka baris perintah pada sistem yang meng-host server web. Pantau perubahan yang dibuat pada *file* yang mungkin membuat server web menjadi *backdoor* dengan web *shell* untuk membuat akses terus-menerus ke sistem. Memantau data jaringan untuk aliran data yang tidak biasa. Proses yang menggunakan jaringan yang biasanya tidak memiliki komunikasi jaringan atau tidak pernah terlihat sebelumnya patut dicurigai. (Corporation, 2015-2025).

2.3 *Broken Access Control*

Broken Access Control merupakan kerentanan keamanan signifikan yang terjadi saat aplikasi atau sistem gagal memberlakukan pembatasan yang tepat pada akses pengguna ke data dan fungsi sensitif. Kerentanan ini dapat menyebabkan akses tidak sah, pelanggaran data, dan memungkinkan pengguna yang tidak sah untuk melewati langkah-langkah keamanan, yang berpotensi menyebabkan konsekuensi parah seperti pelanggaran data dan kerugian finansial. (Elaf Almushiti, 2024). Kerentanan kontrol akses yang umum meliputi: Pelanggaran terhadap prinsip *least privilege* atau menolak secara default, di mana akses seharusnya hanya diberikan untuk kemampuan, peran, atau pengguna tertentu, tetapi tersedia untuk siapa saja. Mengakses *API* dengan kontrol akses yang tidak ada untuk *POST, PUT*, dan *DELETE*. Mengizinkan melihat atau mengedit akun orang lain, dengan memberikan pengenal uniknya menggunakan referensi objek langsung yang tidak aman. (OWASP, 2021). Pengguna dapat mengakses data atau fungsi yang bukan haknya contohnya, user biasa bisa melihat, mengedit, atau menghapus data milik user lain hanya dengan mengganti parameter *ID* pada *URL* atau *request*. Tidak ada pengecekan otorisasi saat mengakses objek atau *endpoint*. Aplikasi tidak

melakukan validasi apakah user benar-benar berhak mengakses data atau fitur tertentu, misalnya halaman admin bisa diakses siapa saja yang tahu *URL*-nya. Pengguna dengan hak akses rendah seperti *user* biasa bisa mendapatkan hak akses lebih tinggi seperti admin tanpa proses otorisasi yang benar, baik secara vertikal (naik peran) maupun horizontal (akses data user lain). *User* dapat mengakses objek internal (*file*, data, *record*) hanya dengan memodifikasi parameter pada *URL* atau request tanpa pemeriksaan hak akses. Penyerang bisa mengakses *endpoint* tersembunyi hanya dengan menebak atau mencoba-coba *URL*, tanpa harus login atau tanpa otorisasi khusus. Pengubahan parameter pada *URL, body request*, atau *cookie* dapat digunakan untuk mendapatkan akses ke data atau fungsi yang seharusnya tidak boleh diakses. Misalnya, endpoint *API* yang seharusnya hanya untuk admin ternyata bisa diakses semua user karena salah konfigurasi. (Magelang, 2024). Kontrol izin menentukan tingkat akses pengguna ke semua aplikasi dan membatasi izin mereka terhadap sumber daya. Akses tidak sah ke aplikasi web terjadi akibat sesi yang tidak valid dalam kode aplikasi web. Contoh kode *PHP* pada gambar 2.1 di bawah ini:


```php
<?php session_start (); ?>

// Checking For Access Validity
if(!$_SESSION['member']) {

    header('Location:login.p
    hp'); exit;

}

// Control Access in
required herr. # Include
'inc/config.php';

# Include 'inc/conn.php';

// Not secure Database Connections for
     accessing Data

$del1 = mysql_query("DELETE FROM close_bid
     WHERE item.name = '" . $item_name .
     "'");

if($delete1) {
    mysql_close($conn);

}

?>
```

Gambar 2.1 *PHP code vulnerability: Broken access control*
(Sumber : Sadeeq, 2025:2)

Gambar 2.1 menunjukkan kerentanan kode *PHP* dalam kontrol akses. Pemeriksaan sesi untuk "member" dilewati setelah pemeriksaan awal, memungkinkan akses tidak sah ke pengguna. Penggunaan "*mysqlquery*" tanpa

sanitasi parameter yang tepat membuat aplikasi rentan terhadap serangan injeksi *SQL*. Sesi baru dibuat pada baris ketiga kode *if(!$_SESSION['member'])* memeriksa apakah pengguna telah login dengan memverifikasi keberadaan dan nilai dari variabel sesi *$_SESSION['member']* ini. Setelah otentikasi, pengguna dapat mengakses halaman "*login.php*". Baris 8 kode *required herr. # Include* berisi informasi konfigurasi tanpa batasan. Pada baris 11 kode *Sdel1 = mysql_query("DELETE FROM close_bid WHERE item.name = '". $item_name .* " "); , pengguna dapat menghapus atau mengubah informasi tanpa izin. Penyerang sering menetapkan nilai *ID* Cookie "*SuperAdmin*" ke *ID* akun administrator (misalnya, ID = 1) dan kemudian mengubah nilai admin di Bagian 2, 3, 4,dll. Saat menjelajahi *file cookie*, pengguna/penyerang dapat mengubah *ID*; misalnya, nilai *ID* dapat berubah dari *ID*=1009 menjadi *ID*=1. Jika sesi tidak didefinisikan dengan benar di halaman admin, hal ini dapat menyebabkan perubahan besar pada akun, sehingga semua izin diberikan kepada *ID*=1 (*MainAdmin*). Ditetapkan bahwa jumlah aplikasi web yang terpengaruh oleh *BAC* dalam sampel adalah 129, dengan 4 faktor risiko utama. Risiko-risiko ini diklasifikasikan berdasarkan pentingnya beberapa variabel independen, yang merupakan penyebab utama kerentanan kontrol akses (*BAC*): data sensitif, pengalihan yang tidak biasa, konfigurasi sesi yang salah, penggunaan bahasa, sistem operasi, dan server atau platform. Semua informasi yang tercantum telah diverifikasi oleh Pusat Keamanan Internet *DIU*. (Sadeeq Jan1, 2025)

2.4 Web Server

Server web adalah komponen penting dari Internet, yang bertindak sebagai komputer yang mengirimkan berbagai jenis informasi kepada pengguna, termasuk teks, audio, dan video. Alat bantu analisis *file log* mencerminkan sifat multi disiplin dari manajemen web server dan sangat penting untuk memantau kinerja server dan memecahkan masalah. (Prajkta khaire, 2022). Web server berfungsi melayani permintaan *HTTP* atau *HTTPS* pada *port* 80 dan 443, mengirimkan *file* atau data dari direktori tertentu sebagai respons ke klien. *Port* 80 digunakan untuk *HTTP*, sedangkan *port* 443 untuk *HTTPS* yang lebih aman. (Lau, 2021). Pengguna memasukkan *URL* website di browser, yang kemudian mengirimkan permintaan

HTTP atau *HTTPS* ke web server. Web server menerima permintaan tersebut dan mencari *file* atau data yang diminta di server. Jika data ditemukan, web server mengirimkan kembali *file* website kepada browser dalam bentuk *HTTP* response agar dapat ditampilkan. Jika data tidak ditemukan, web server mengirimkan kode *error* seperti 404 *Not Found* atau 403 *Forbidden*.

Kode status respons *HTTP* menunjukkan apakah permintaan *HTTP* tertentu telah berhasil diselesaikan. (Docs, 2025). Respons dikelompokkan dalam lima kelas yaitu *Informational responses* (100 – 103), *Successful responses* (200 – 226), *Redirection messages* (300 – 308), *Client error responses* (400 – 451), *Server error responses* (500 – 511). Kode yang lengkap disebutkan dalam bab ini dapat dilihat pada bagian lampiran 1. Berikut seperti pada tabel 2.2:

Tabel 2.2 Kode status respons *HTTP*

No	Keterangan	Penjelasan
1	Informational responses	100 Continue: Respons sementara ini mengindikasikan bahwa klien harus melanjutkan permintaan atau mengabaikan respons jika permintaan sudah selesai. 101 Switching Protocols: Kode ini dikirim sebagai respons terhadap header permintaan Upgrade dari klien dan menunjukkan protokol yang dialihkan oleh server. 102 Processing: Kode ini digunakan dalam konteks WebDAV untuk menunjukkan bahwa permintaan telah diterima oleh server, tetapi tidak ada status yang tersedia pada saat respons. 103 Early Hints: Kode status ini terutama ditujukan untuk digunakan dengan header Tautan, memungkinkan agen pengguna mulai memuat sumber daya terlebih dahulu saat server menyiapkan respons atau melakukan pra-koneksi ke asal dari mana halaman akan membutuhkan sumber daya.
2	Successful responses	200 Ok: Permintaan berhasil. Hasil dan arti dari "berhasil" bergantung pada metode HTTP.

No	Keterangan	Penjelasan
		201 Created: Permintaan berhasil, dan sumber daya baru telah dibuat sebagai hasilnya. Ini biasanya merupakan respons yang dikirim setelah permintaan POST, atau beberapa permintaan PUT. 202 Accepted: Permintaan telah diterima tetapi belum ditindaklanjuti. Ini tidak pasti, karena tidak ada cara dalam HTTP untuk kemudian mengirim respons asinkron yang menunjukkan hasil permintaan. Ini dimaksudkan untuk kasus di mana proses atau server lain menangani permintaan, atau untuk pemrosesan batch. 203 Non-Authoritative Information: Kode respons ini berarti metadata yang dikembalikan tidak persis sama dengan yang tersedia dari server asal, tetapi dikumpulkan dari salinan lokal atau pihak ketiga. Ini sebagian besar digunakan untuk mirror atau cadangan sumber daya lain. Kecuali untuk kasus khusus itu, respons 200 OK lebih disukai daripada status ini. 204 No Content: Tidak ada konten untuk dikirim untuk permintaan ini, tetapi tajuknya berguna. Agen pengguna dapat memperbarui tajuk yang di-cache untuk sumber daya ini dengan yang baru. 205 Reset Content: Memberi tahu agen pengguna untuk mengatur ulang dokumen yang mengirim permintaan ini.
3	Redirection messages	300 Multiple Choices: Dalam negosiasi konten yang digerakkan oleh agen, permintaan memiliki lebih dari satu kemungkinan respons dan agen pengguna atau pengguna harus memilih salah satunya. Tidak ada cara standar bagi klien untuk secara otomatis memilih salah satu respons, jadi ini jarang digunakan. 301 Moved Permanently: URL sumber daya yang diminta telah diubah secara permanen. URL baru diberikan dalam respons.

No	Keterangan	Penjelasan
		302 Found: Kode respons ini berarti bahwa URI sumber daya yang diminta telah diubah sementara. Perubahan lebih lanjut pada URI mungkin dilakukan di masa mendatang, jadi URI yang sama harus digunakan oleh klien dalam permintaan di masa mendatang. 303 See Other: Server mengirimkan respons ini untuk mengarahkan klien mendapatkan sumber daya yang diminta di URI lain dengan permintaan GET. 304 Not Modified: Ini digunakan untuk tujuan penyimpanan cache. Ini memberitahu klien bahwa respons belum dimodifikasi, sehingga klien dapat terus menggunakan versi respons yang sama yang di-cache. 305 Use Proxy: Didefinisikan dalam versi spesifikasi HTTP sebelumnya untuk menunjukkan bahwa respons yang diminta harus diakses oleh proksi. Ini telah usang karena masalah keamanan terkait konfigurasi in-band dari proksi.
4	Client error responses	400 Bad Request: Server tidak dapat atau tidak akan memproses permintaan karena sesuatu yang dianggap sebagai kesalahan klien (misalnya, sintaks permintaan yang salah, pembingkaian pesan permintaan yang tidak valid, atau perutean permintaan yang menipu). 401 Unauthorized: Meskipun standar HTTP menetapkan "unauthorized", secara semantik respons ini berarti "unauthenticated". Yaitu, klien harus mengautentikasi dirinya sendiri untuk mendapatkan respons yang diminta. 402 Payment Required: Tujuan awal kode ini adalah untuk sistem pembayaran digital, namun kode status ini jarang digunakan dan tidak ada konvensi standar yang ada. 403 Forbidden: Klien tidak memiliki hak akses ke konten; yaitu, tidak sah, jadi server menolak untuk memberikan sumber daya yang diminta.

No	Keterangan	Penjelasan
		Tidak seperti 401 Unauthorized, identitas klien diketahui oleh server. 404 Not Found: Server tidak dapat menemukan sumber daya yang diminta. Dalam peramban, ini berarti URL tidak dikenali. Dalam API, ini juga bisa berarti bahwa titik akhir valid tetapi sumber daya itu sendiri tidak ada. Server juga dapat mengirim respons ini alih-alih 403 Forbidden untuk menyembunyikan keberadaan sumber daya dari klien yang tidak berwenang. Kode respons ini mungkin yang paling terkenal karena seringnya terjadi di web. 405 Method Not Allowed: Metode permintaan dikenal oleh server tetapi tidak didukung oleh sumber daya target. Misalnya, sebuah API mungkin tidak mengizinkan DELETE pada sebuah sumber daya, atau metode TRACE sama sekali.
5	Server error responses	500 Internal Server Error: Server telah menemukan situasi yang tidak tahu bagaimana menanganinya. Kesalahan ini bersifat umum, menunjukkan bahwa server tidak dapat menemukan kode status 5XX yang lebih sesuai untuk merespons. 501 Not Implemented: Metode permintaan tidak didukung oleh server dan tidak dapat ditangani. Satu-satunya metode yang wajib didukung oleh server (dan karenanya tidak boleh mengembalikan kode ini) adalah GET dan HEAD. 502 Bad Gateway: Respons kesalahan ini berarti bahwa server, saat berfungsi sebagai gateway untuk mendapatkan respons yang diperlukan untuk menangani permintaan, menerima respons yang tidak valid. 503 Service Unavailable: Server tidak siap untuk menangani permintaan. Penyebab umum adalah server yang sedang dalam perbaikan atau kelebihan beban. Perhatikan bahwa bersama dengan respons ini, halaman yang mudah digunakan yang menjelaskan masalah tersebut

No	Keterangan	Penjelasan
		harus dikirim. Respons ini harus digunakan untuk kondisi sementara dan tajuk HTTP Retry-After, jika memungkinkan, harus berisi perkiraan waktu sebelum pemulihan layanan. Webmaster juga harus memperhatikan tajuk terkait caching yang dikirim bersama dengan respons ini, karena respons kondisi sementara ini biasanya tidak boleh di-cache. 504 Gateway Timeout: Respons kesalahan ini diberikan ketika server bertindak sebagai gateway dan tidak bisa mendapatkan respons tepat waktu. 505 HTTP Version Not Supported: Versi HTTP yang digunakan dalam permintaan tidak didukung oleh server.

Web server statis hanya mengirimkan *file* website apa adanya tanpa perubahan sedangkan Web server dinamis, dilengkapi dengan database dan server aplikasi untuk menghasilkan konten yang dapat diperbarui secara dinamis sebelum dikirim ke browser. (A., 2024)

2.5 *Digital Forensic*

Forensik Digital merupakan bidang penting dalam keamanan siber yang berfokus pada investigasi dan interpretasi bukti digital yang terkait dengan kejahatan siber. Metode forensik digital berguna untuk berbagai alasan, termasuk menyelidiki kejahatan dan kesalahan internal, merekonstruksi insiden keamanan, memecahkan masalah operasional, dan memulihkan dari kegagalan sistem yang tidak disengaja. Forensik Digital tidak memiliki prosedur teknis khusus yang akan memberikan instruksi langkah demi langkah yang akan mengarahkan analis ke jawaban. Diperlukan banyak jam pelatihan dan praktik untuk mencapai tingkat kemampuan ahli. (Salfati, 2022). Aspek utama dalam digital *forensic* adalah identifikasi dan pengumpulan bukti digital dari berbagai sumber seperti komputer, server, perangkat *mobile*, jaringan, dan media penyimpanan. Analisis data untuk menemukan jejak digital, seperti *file* yang dihapus, *log* aktivitas, *malware*, akses tidak sah, atau perubahan data. Pelestarian bukti agar tetap utuh dan tidak

terkontaminasi selama proses investigasi. Pembuatan laporan forensik yang mendokumentasikan temuan secara detail dan dapat digunakan dalam proses hukum. Penerapan di dalam digital *forensic* yaitu investigasi akses tidak sah dan pencurian data pada server perusahaan. Analisis serangan siber pada *website* atau aplikasi web. Penelusuran *malware* dan *spyware* yang tersembunyi dalam sistem. (Reddy, 2021)

2.6 *Log*

Log adalah rekaman informasi yang dibuat oleh sistem, aplikasi, atau perangkat yang melacak peristiwa dan aktivitas dari waktu ke waktu. Ini sangat penting dalam banyak bidang, seperti *TI*, pengeboran, dan analisis data. Penelitian analisis *log* otomatis bertujuan untuk menggunakan *log* perangkat lunak secara efisien untuk rekayasa keandalan, memungkinkan deteksi *anomali*, prediksi kegagalan, dan diagnosis. (Shilin He, 2020). Analisis *log* telah maju dalam beberapa tahun terakhir, tetapi tantangan tetap ada dalam menganalisis data bervolume besar dan mengidentifikasi masalah keamanan dalam sistem perangkat lunak dan sistem *IoT.* (J. Svacina, 2020). Peran *Log* dalam Digital Forensik yaitu menjadi dasar deteksi anomali dan pola tidak wajar yang mengindikasikan adanya pelanggaran keamanan atau aktivitas berbahaya. Teknik Analisis *Log* dalam Digital Forensik ialah Data *Collection*, mengumpulkan *log* dari berbagai sumber (server, perangkat jaringan, aplikasi) dengan menjaga integritas data agar tidak dimanipulasi. *Log Parsing* dengan cara mengubah data mentah *log* menjadi format yang mudah dibaca dan dianalisis, menyoroti detail penting seperti *timestamp, IP address*, dan aksi pengguna. *Event Correlation* yaitu menghubungkan peristiwa dari berbagai *log* untuk mendapatkan gambaran utuh tentang kejadian, biasanya menggunakan *SIEM* (*Security Information and Event Management*). *Timeline Construction* untuk menyusun urutan waktu kejadian berdasarkan log untuk merekonstruksi insiden secara kronologis. *Anomaly Detection* yaitu mengidentifikasi pola atau aktivitas yang tidak biasa, seperti lonjakan aktivitas, akses ilegal, atau transfer data mencurigakan, seringkali menggunakan *machine learning* atau *AI* untuk meningkatkan akurasi deteksi. *Keyword Search* untuk mencari entri tertentu dalam *log* menggunakan kata kunci spesifik untuk mempercepat proses investigasi.

Automated Analysis Tools digunakan sebagai alat otomatis seperti *Splunk, ELK Stack,* atau *Graylog* untuk memproses dan menganalisis *log* dalam jumlah besar secara efisien. *Visual Data Representation* digunakan untuk memvisualisasikan data *log* dengan grafik, dashboard, atau *heat map* untuk memudahkan identifikasi pola dan anomaly Selama proses pengumpulan dan analisis, menjaga integritas *log* sangat krusial agar bukti tetap sah dan dapat dipertanggungjawabkan secara hukum. Setiap perubahan atau manipulasi pada *log* dapat mengurangi validitas hasil investigasi. *Log* merupakan fondasi utama dalam investigasi digital forensik karena menyediakan bukti objektif yang dapat digunakan untuk mendeteksi, menganalisis, dan membuktikan insiden keamanan secara ilmiah dan sistematis, (Salvationdata, 2024).

2.7 *Access Log*

Access Log adalah komponen penting dari manajemen server *web*, berfungsi sebagai catatan komprehensif interaksi pengguna dengan situs *web*. Ini menangkap data penting seperti permintaan klien, respons server, dan perilaku pengguna, yang dapat dimanfaatkan untuk berbagai tujuan analitis. Bagian berikut menguraikan signifikansi, solusi penyimpanan, dan masalah privasi yang terkait dengan *log* akses. *Access Log* memberikan wawasan tentang perilaku pengguna, memungkinkan administrator situs *web* untuk mengidentifikasi pola dan mengoptimalkan pengalaman pengguna. (Yogi, 2019). Access *Log* dapat secara tidak sengaja mengekspos informasi pengguna yang sensitif, termasuk alamat *IP* dan detail perangkat, meningkatkan risiko privasi yang signifikan. (Kamil Kaczyński, 2020).

Access log biasanya mencatat beberapa data penting, antara lain yaitu tanggal dan waktu akses (*timestamp*), Alamat *IP* atau *hostname* klien, *Username* (jika autentikasi digunakan), Metode permintaan (*GET, POST*, dsb.), *URL* atau sumber daya yang diakses, Status respons server (kode *HTTP* seperti 200, 404, 500), Ukuran respons, *User agent* (informasi tentang browser atau aplikasi klien), *Referrer* (situs asal sebelum mengakses sumber daya). Contoh entri *access log* web server 127.0.0.1 – – [31/Jan/2021:16:09:05 -0400] "*GET /admin/ HTTP/1.1" 200 "-" "Mozilla/5.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322)".*

(Srinivas, 2021). Fungsi dan pentingnya *Access Log* untuk memantau siapa yang mengakses sistem, kapan, dan apa yang diakses. Ini penting untuk keamanan dan kepatuhan regulasi, mengidentifikasi sumber masalah, seperti *error* 404 (halaman tidak ditemukan) atau lonjakan trafik yang tidak wajar, menganalisis pola akses untuk mengidentifikasi halaman yang lambat atau sering diakses, sehingga dapat dilakukan optimasi, mengidentifikasi aktivitas mencurigakan, seperti *brute force attack, scraping*, atau serangan *DDoS* berdasarkan pola akses yang tidak biasa. Menjadi sumber bukti utama dalam investigasi insiden keamanan, karena dapat merekonstruksi kronologi kejadian dan mengidentifikasi pelaku atau metode serangan. (SentinelOne, 2025)

2.8 *Machine Learning*

Machine learning adalah suatu metode yang membuat komputer memiliki kemampuan dalam mempelajari dan melakukan sebuah pekerjaan secara otomatis. Proses *machine learning* ini didasari oleh gagasan bahwa mesin dapat belajar sendiri tanpa harus diprogram secara eksplisit. Teknologi ini semakin penting di berbagai bidang, termasuk analisis data besar, pemrosesan bahasa alami, dan pengenalan gambar. Inovasi dalam pembelajaran mesin, seperti jaringan saraf konvolusional (*CNN*), meningkatkan kemampuan di bidang-bidang seperti pengenalan gambar dan ucapan. (Sim Sang Gyoo, 2019). Algoritma pembelajaran mesin membantu komputer melakukan tugas secara lebih efisien dan efektif, mengubah industri dan kehidupan kita sehari-hari. (Kawahara, 2020).

Machine learning (ML), sebagai bagian dari kecerdasan buatan (*AI*), semakin berperan penting dalam meningkatkan efisiensi dan akurasi investigasi forensik digital . Dalam forensik digital, *ML* melibatkan pengembangan model dan algoritma komputer yang dapat menganalisis dan menafsirkan bukti digital dalam jumlah besar tanpa perlu pemrograman eksplisit . Peran *Machine Learning* dalam Forensik Digital. Penerapan *ML* dalam forensik digital masih dalam tahap awal namun menunjukkan potensi transformatif . Ini memungkinkan investigator untuk mengatasi volume data yang besar dengan meningkatnya kejahatan siber dan volume data digital, *ML* membantu menganalisis kumpulan data yang luas dan kompleks yang tersimpan di berbagai lingkungan komputasi awan dan jaringan.

Deteksi pola dan anomali algoritma *ML* dapat mengidentifikasi pola tersembunyi dan perilaku anomali yang mungkin mengindikasikan aktivitas kriminal atau serangan siber. *ML* dapat membantu dalam merekonstruksi kronologi kejadian kejahatan siber, yang seringkali menjadi tantangan karena kompleksitas dan volume data. *ML* digunakan untuk mengklasifikasikan konten media penyimpanan, memulihkan *file* yang dihapus, dan mengekstrak fitur penting dari bukti digital. *ML* membantu investigator melakukan investigasi yang lebih efektif dan efisien dengan mengotomatiskan tugas-tugas analisis data yang repetitif. (Yusra Al Balushi(B), 2023).

2.9 *SPL (Search Processing Languange)*

Sebagai alat canggih yang digunakan terutama dalam lingkup *Splunk* untuk operasi pencarian dan analisis data, *SPL* (*Search Processing Language*) memungkinkan pengguna untuk menanyakan, menganalisis, dan memvisualisasikan kumpulan data yang sangat besar. *SPL (Search Processing Language)* dirancang untuk menangani berbagai tipe data dan menyediakan serangkaian perintah yang kaya untuk manipulasi data dan pelaporan. (Mehta, 2021). *SPL (Search Processing Language)* memungkinkan pengguna untuk melakukan kueri kompleks pada data yang diindeks, memfasilitasi analisis data *real-time*. Untuk mengukur efektivitas *Search Processing Language (SPL)* , waktu yang dibutuhkan *Search Processing Language(SPL)* untuk menjalankan pencarian yang efektif biasanya memiliki waktu eksekusi yang singkat. Waktu eksekusi terjadwal berikutnya dari pencarian terjadwal dikendalikan oleh atribut *realtime_schedule*. Jika nilai ini diatur ke 1, penjadwal mendasarkan penentuan waktu eksekusi pencarian terjadwal berikutnya pada waktu saat ini. Jika diatur ke 0, penjadwal mendasarkan penentuan waktu eksekusi pencarian terjadwal berikutnya pada waktu eksekusi pencarian terakhir. Ini dikenal sebagai penjadwalan berkelanjutan *continuous scheduling*. Dalam penjadwalan berkelanjutan, penjadwal tidak pernah melewati periode eksekusi terjadwal, meskipun eksekusi pencarian yang disimpan mungkin tertinggal tergantung pada beban penjadwal. Metode penjadwalan ini direkomendasikan saat mengaktifkan opsi indeks ringkasan *summary index*. Ini memastikan penjadwal mengeksekusi pencarian pada rentang

waktu terbaru dengan memprioritaskan pencarian dengan *realtime_schedule* diatur ke 1 daripada yang menggunakan penjadwalan berkelanjutan *realtime_schedule = 0*. (Splunk, Splunk Cloud Platform Admin Manual, 2025).

Perintah *SPL* terdiri dari argumen wajib dan opsional. Argumen wajib ditunjukkan dalam kurung sudut < >. Argumen opsional dilampirkan dalam kurung siku [ ]. *Syntax bin [<bins-options>...] <field> [AS <newfield>]*. Argumen yang diperlukan adalah *<field>*. Untuk menggunakan perintah ini, minimal harus menentukan bin *<field>*. Argumen opsional adalah *[<bins-options>...]* dan *[AS <newfield>]*. Banyak perintah menggunakan kata kunci dengan beberapa argumen atau opsi. Contoh kata kunci meliputi *AS, BY, OVER, WHERE*. Sintaks dalam dokumentasi *Splunk* menggunakan huruf besar pada semua kata kunci. Saat operator logika disertakan dalam sintaks perintah, harus selalu menentukan operator dalam huruf besar. Operator logika meliputi *AND, OR, NOT, XOR*. Perintah pencarian mengevaluasi operator logika dalam urutan preseden yang berbeda dari perintah *eval* dan *where*. Untuk menjelaskan kepada pengguna bagaimana cara menulis perintah atau query dengan benar, (Splunk, Understanding SPL syntax, 2025). Khususnya mengenai jenis nilai atau field yang diharapkan untuk setiap bagian dari sintaks seperti pada tabel 2.3:

Tabel 2.3 Tipe data

Sintaks	Tipe Data	Catatan
<bool>	*Boolean*	Gunakan *true* atau *false*. Variasi lain diterima. Misalnya, untuk *true* juga bisa menggunakan 'T', 't', *'TRUE'*, '*yes*', atau angka satu (1). Untuk *false* Anda bisa juga menggunakan '*no*', angka nol (0), dan variasi dari kata *false*, mirip dengan variasi dari kata *true*.
<field>	Nama *field*	Nama *field*. Tidak bisa menentukan karakter *wildcard* untuk nama *field*.
<int> atau *<integer>*	*Integer*	Bilangan bulat (*integer*) yang bisa bernilai positif atau negatif. Terkadang

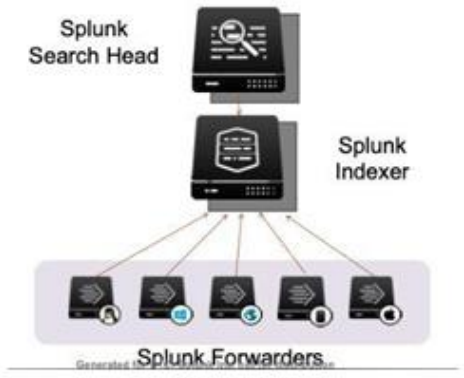
Sintaks	Tipe Data	Catatan
		disebut sebagai integer "bertanda" (*signed integer*). Lihat <*unsigned int*>.
<*string*>	*String*	Lihat <*wc-string*>.
<*unsigned int*>	*unsigned integer*	Bilangan bulat tak bertanda (*unsigned integer*) harus bernilai positif. *Unsigned integer* bisa memiliki nilai yang lebih besar daripada *signed integer*.
<*wc-field*>	Nama *field* atau sebagian nama dengan karakter *wildcard* untuk menentukan beberapa *field* dengan nama yang mirip.	Gunakan karakter asterisk (*) sebagai karakter *wildcard*.
<*wc-string*>	Nilai *string* atau sebagian nilai *string* dengan karakter *wildcard*.	Gunakan karakter asterisk (*) sebagai karakter *wildcard*.

Klausa <*by-clause*> dan <*split-by-clause*> bukanlah argumen yang sama. Ketika menggunakan <*by-clause*>, satu baris dikembalikan untuk setiap nilai berbeda dari bidang <*by-clause*>. Sebuah <by-clause> menampilkan setiap item unik dalam baris terpisah. Anggap <by-clause> sebagai pengelompokan. <*split-by-clause*> menampilkan setiap item unik dalam kolom terpisah. Anggap <*split-by-clause*> sebagai pemisahan atau pembagian. Karakter *wildcard* (*) tidak diterima dalam klausa *BY*. Untuk perintah yang lebih lengkap disebutkan dalam bab ini dapat dilihat pada bagian lampiran 2.

2.10 *Splunk*

Splunk adalah alat analisis *data* yang canggih yang digunakan untuk menangkap, mencari, dan menganalisis *data log* secara *real-time*, menjadikannya

sangat berharga untuk wawasan keamanan dan operasional. *Splunk* unggul dalam menangani kumpulan data besar, memungkinkan pengguna menemukan pola dan menemukan *anomali* dengan menggunakan bahasa kueri yang kuat**.** *Splunk* berfungsi melalui 3 alat penyusun inti, seperti pada gambar 2.2 :



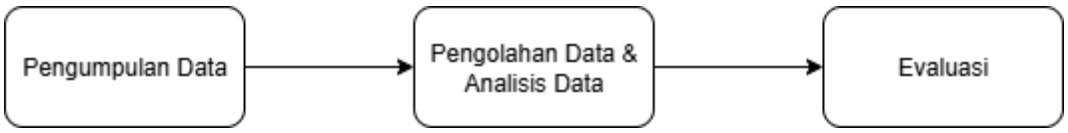
Gambar 2.2 Cara Kerja *Splunk*
(Sumber : Suheni, 2023:2)

Dari gambar 2.2 merupakan bagian cara kerja *splunk* yang pertama ialah setiap perangkat yang menghasilkan data dipasang di alat bernama *Splunk Forwarders* yang berfungsi untuk mengumpulkan data dari perangkat serta memfilter dan mengarahkan data ke penerima tertentu berdasarkan sumber, tipe sumber, atau pola dalam peristiwa itu sendiri lalu mengirimnya menuju *Splunk Indexer* yang merupakan tempat berkumpulnya data dari berbagai perangkat. Untuk kebutuhan tingkat atas yaitu berkaitan dengan klien dan visualisasi maka alat yang berfungsi adalah *Splunk Search Head*. *Splunk* juga memiliki fitur penyaringan, pencarian, masukan, modifikasi, pelaporan, dan penghapusan data. (Suheni, 2023)

Splunk mengumpulkan *data log* dari berbagai sumber seperti perangkat jaringan, *server*, aplikasi, dan sistem lainnya secara *real-time*. Data ini kemudian diproses dan di indeks sehingga mudah dicari dan dianalisis dengan bantuan *AI* dan *machine learning*, sehingga melakukan pencarian cepat dengan *query* yang kompleks, memantau kejadian secara langsung, memberikan peringatan otomatis, serta mengintegrasikan dan mengotomatiskan respons keamanan untuk mendeteksi dan menangani ancaman siber secara efektif. (Jason Wiharja, 2024)

BAB III
METODOLOGI PENELITIAN

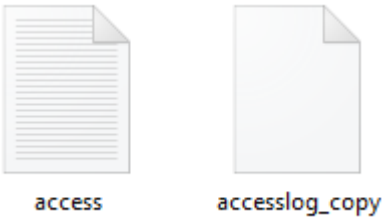
Metode penelitian yang digunakan pada penelitian ini dimulai dengan pengumpulan data, pengolahan data dan analisis data serta evaluasi dengan diagram alir sebagai berikut.



Gambar 3.1 Tahapan Penelitian

3.1 Pengumpulan Data

Pada tahap ini, mengumpulkan data yang diperlukan untuk penelitian berupa informasi dari survei, wawancara, eksperimen, atau dokumen lainnya. Tujuannya adalah mendapatkan data yang relevan dan cukup untuk mendukung penelitian yang sedang dilakukan. Dengan cara mengamankan bukti berupa data *acces log*, *error log*, *syslog*, dan *auth log* dengan menggunakan tool *SSH* dan *SCP* yang akan digunakan untuk di analisa *log* serangannya. Data *log* yang digunakan dipastikan tidak ada kesalahan pada saat di kumpulkan serta sudah dipastikan kualitas dan integritas data yang akurat serta melakukan *backup* data untuk mencegah kehilangan data. Berikut salah satu contoh data *acces log* yang sudah diamankan.

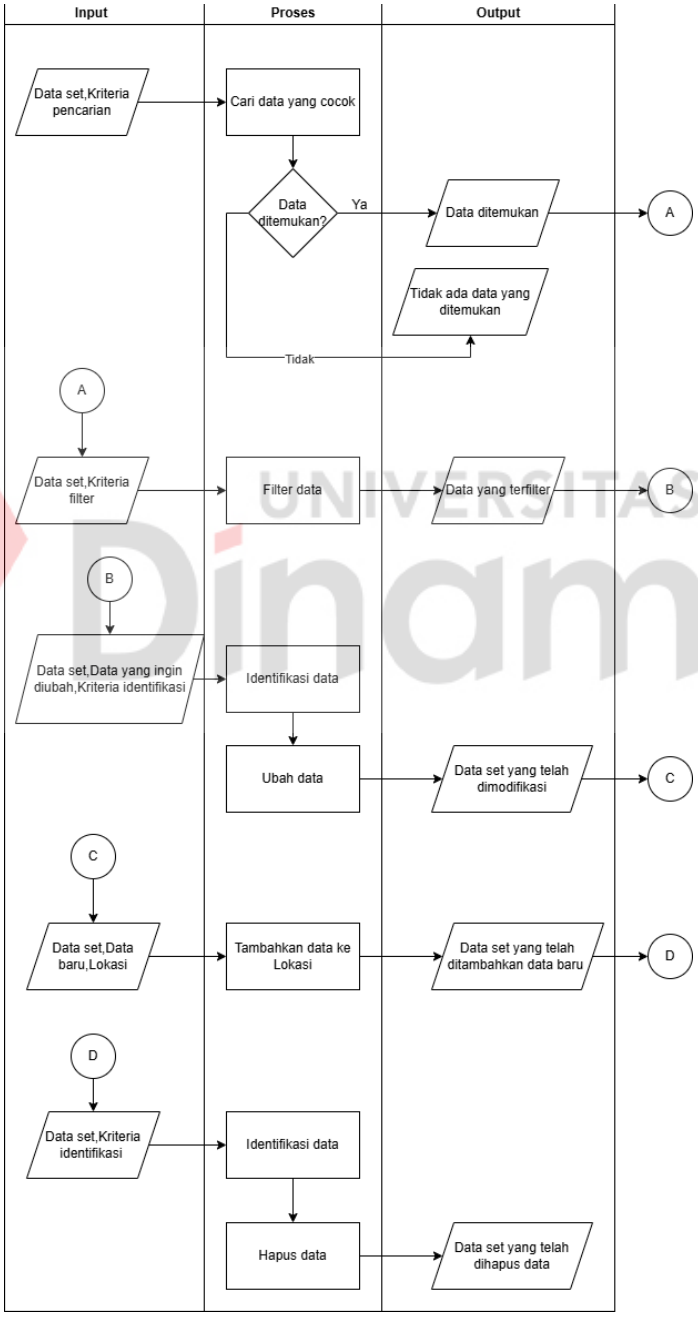


Gambar 3.2 Data *Log access*

3.2 Pengolahan Data

Setelah data dikumpulkan, langkah berikutnya adalah memeriksa data tersebut. Ini melibatkan pengecekan keakuratan dan kelengkapan data. Dipastikan

bahwa data yang dikumpulkan tidak memiliki kesalahan dan siap untuk dianalisis lebih lanjut. Untuk mengecek keakuratan dan kelengkapan data, dilakukan dengan mendeteksi serangan *Backdoor Shell* dan *Broken Access Control* dengan cara mengeksekusi data *access log* yang telah didapatkan menggunakan *search processing language* yang diterapkan untuk mengidentifikasi dan mengekstrak informasi yang relevan dari data yang dikumpulkan.



Gambar 3.3 Cara kerja sistem filter berbasis *Splunk*

Tahap ini adalah proses hasil dari *Splunk* untuk dianalisis lebih lanjut. Berikut teknik yang digunakan saat proses pengolahan data:

a) *Data searching*: Untuk menemukan data spesifik dalam kumpulan data yang lebih besar berdasarkan kriteria pencarian tertentu. Ini dilakukan menggunakan kueri pencarian yang menentukan kondisi yang harus dipenuhi oleh data. Perintah dasar "*search"* digunakan untuk menemukan event tertentu berdasarkan kata kunci atau pola yang ditentukan. Untuk mengukur efektivitas bisa menggunakan *search* mode yaitu mode *fast, smart, verbose.* Tergantung pada mode yang diatur saat menjalankan pencarian, dapat melihat semua data yang tersedia untuk pencarian, tetapi dengan mengorbankan waktu pencarian yang lebih lama, dapat dipercepat dan dirampingkan dengan mode tertentu.

b) *Data filtering*: Untuk memilih subset data dari kumpulan data yang lebih besar berdasarkan kriteria tertentu. Ini melibatkan penggunaan operasi penyaringan dalam kueri atau menggunakan perintah penyaringan khusus. Dengan menggunakan perintah seperti "*where"* seperti *index=logs | where status="error"*

c) *Data modification:* Untuk mengubah atau memperbarui data yang ada. Di dalam *SPL* menawarkan perintah atau fungsi untuk memodifikasi data berdasarkan kriteria atau lokasi tertentu. Ini dapat dilakukan dengan menggunakan perintah seperti "*eval*", *index=logs | eval response_time_ms=response_time*1000*.

d) *Data insertion:* Untuk menambahkan data baru ke dalam set data. Perintah atau fungsi khusus digunakan untuk menambahkan data baru ke dataset yang ada.

e) *Data deletion:* Untuk menghapus data tertentu dari set data. Perintah atau fungsi disediakan untuk menghapus data yang memenuhi kriteria tertentu.

Setelah melakukan melakukan *parsing* untuk memecah *log* menjadi *event,* lalu mengekstrak *timestamp*, *host*, *source*, *sourcetype*, dan *field* lain yang relevan serta menyimpan hasil *parsing* ke dalam *index* untuk mempercepat pencarian, data akan disimpan dalam format *time-series* agar mudah dianalisis secara kronologis. Proses dari tahap *examination* yaitu untuk mencari *string* atau pola yang

mengindikasikan aktivitas mencurigakan, seperti perintah *shell* yang tidak biasa, akses *file* yang tidak sah, atau penggunaan fungsi berbahaya. Serta mencari perubahan yang signifikan dalam perilaku *server*, seperti peningkatan jumlah permintaan *HTTP*, atau perubahan pola akses *file*. Selain itu untuk mencari aktivitas yang terjadi dalam rentang waktu yang mencurigakan, seperti peningkatan jumlah permintaan *HTTP* secara tiba-tiba, atau perubahan konfigurasi yang terjadi sebelum aktivitas mencurigakan. Lalu untuk *pseudocode* dari data *examination* adalah sebagai berikut:

Berikut adalah pseudocode singkat dari diagram alir yang diberikan:

1. *Searching* Data

   function CariData(dataSet, kriteriaPencarian):

   Cari data yang cocok dalam dataSet berdasarkan kriteriaPencarian

   If data ditemukan:

   Return "Data ditemukan"

   other:

   Return "Tidak ada data yang ditemukan"

2. *Filtering* Data

   function FilterData(dataSet, kriteriaFilter):

   Filter data dari dataSet berdasarkan kriteriaFilter

   Return "Data yang terfilter"

3. *Modification* Data

   function UbahData(dataSet, dataYangInginDiubah, kriteriaIdentifikasi):

   Identifikasi data dalam dataSet berdasarkan kriteriaIdentifikasi

   Ubah data yang teridentifikasi dengan data Yang Ingin Diubah

   Return "Data set yang telah dimodifikasi"

4. *Add* Data

   Function TambahData(dataSet, dataBaru, lokasi):

   Add dataBaru ke dataSet pada lokasi yang ditentukan

   Return "Data set yang telah ditambahkan data baru"

5. *Delete* Data

   Function HapusData(dataSet, kriteriaIdentifikasi):

   Identifikasi data dalam dataSet berdasarkan kriteriaIdentifikasi

Delete data yang teridentifikasi

Return "Data set yang telah dihapus data"

3.3 Analisis Data

Setelah data *access log* diperiksa, dilanjutkan dengan melakukan analisis. Pada tahap ini, data yang telah dikumpulkan dianalisis secara mendalam untuk mencari pola, tren, atau hubungan yang mencurigakan. Tujuan utamanya adalah menemukan bukti yang bisa menjelaskan akar penyebab terjadinya insiden.

Untuk mengidentifikasi adanya serangan *backdoor shell* atau aktivitas mencurigakan serupa, bisa menggunakan *query* seperti pada gambar 3.4 :

```
index=webserver_logs sourcetype=access_combined
(uri_path="*.php" OR uri_path="*.asp" OR uri_path="*.jsp")
method=POST | stats count by uri_path, clientip, useragent |
where count > 5.
```

Gambar 3.4 *Query Backdoor Shell*

Untuk mengidentifikasi potensi serangan Broken Access Control pada fungsi perubahan data evaluasi, dapat menggunakan *query Splunk* pada gambar 3.5 sebagai berikut:

```
sourcetype="access_combined" uri_path="/admin/evaluasi/edit-
save/*"Status=302 NOT user_role="admin" | stats count by
clientip, user, uri_path | sort -count.
```

Gambar 3.5 *Query Broken Access Control*

Untuk menganalisis *log* akses *web* dengan mencari kesalahan yang terjadi dengan menggunakan *query* seperti pada gambar 3.6:

```
sourcetype="access_combind" | where status>=400 | stats count
by status, uri_path | sort -count.
```

Gambar 3.6 *Query Filtering Backdoor Shell*

Untuk mendeteksi upaya akses mencurigakan ke *file* atau direktori sensitif pada *web* server yang menghasilkan kode kesalahan *HTTP 404* (Tidak Ditemukan) atau 403 (Terlarang). Seperti pada gambar 3.7:

```
index=webserver_logs (uri_path="/.*"OR uri_path="/*.ht*")
status=404 OR status=403  | stats count by uri_path, clientip.
```

Gambar 3.7 *Query Filtering Broken Access Control*

Untuk membantu melihat berapa banyak respons dari *web server* yang termasuk dalam kategori cepat, normal, lambat, atau sangat lambat, sehingga dapat memahami kinerja web server secara keseluruhan. Dapat menggunakan *query* seperti pada gambar 3.8:

```
sourcetype="access_combined" | eval response_time_category=
case (response_time<100, "Fast",response_time<500, "Normal",
response_time<1000, "Slow", 1=1, "Very Slow" ) | stats count
by response_time_category.
```

Gambar 3.8 *Respon Query*

Untuk mengelompokkan hasil hitungan berdasarkan kombinasi unik dari *threat_level* yang baru dibuat dan *uri_path* jalur *URI* yang diakses. Dapat menggunakan *query* seperti pada gambar 3.9:

```
index=webserver_logs sourcetype=access_combined | eval threa
t_level=if(match(useragent,".*(curl|wget|python).*"),
"High", "Low") | stats count by threat_level, uri_path.
```

Gambar 3.9 *Query Insertion Backdoor Shell*

Untuk mendeteksi upaya akses ke lokasi atau jalur yang dianggap sensitif. Dapat menggunakan *query* seperti pada gambar 3.10:

```
index=webserver_logs sourcetype=access_combined | eval is_sensitive=if(uri_path="/admin/" OR uri_path="/config/", "Yes", "No") | stats count by is_sensitive, clientip.
```

Gambar 3.10 *Query Insertion Broken Access Control*

Untuk mendapatkan gambaran yang lebih jelas tentang bagaimana pengguna atau script berinteraksi dengan fungsionalitas utama website, bukan hanya elemen visualnya. Dapat menggunakan *query* seperti pada gambar 3.11:

```
index=webserver_logs sourcetype=access_combined | where NOT (uri_path="*.css" OR uri_path="*.js" OR uri_path="*.png") | stats count by method, uri_path.
```

Gambar 3.11 *Query Deletion Backdoor Shell*

Untuk menganalisis *log* akses web server dengan tujuan mengidentifikasi aktivitas yang tidak normal atau berpotensi tidak sah pada akses yang tidak sesuai dengan pola normal, terutama yang berkaitan dengan akses ke area administratif dengan cara menyaring lalu lintas yang sah dan menyoroti kejadian yang memerlukan investigasi lebih lanjut. Dapat menggunakan *query* pada gambar 3.12:

```
index=webserver_logs sourcetype=access_combined | where NOT (status=200 AND user_role="admin") | stats count by uri_path, status.
```

Gambar 3.12 *Query Deletion Broken Access Control*

3.4 Evaluasi

Langkah terakhir adalah melaporkan hasil penelitian. Menyusun laporan yang mencakup semua temuan, analisis, dan kesimpulan. Laporan ini biasanya disusun agar mudah dipahami oleh orang lain, termasuk mereka yang mungkin tidak memiliki latar belakang teknis. Tahap keempat yaitu evaluasi, merangkum hasil

temuan data *log access* yang telah didapatkan dan rekomendasi lebih lanjut. Akhirnya, hasil analisis didokumentasikan dalam laporan komprehensif, merinci temuan dan rekomendasi untuk meningkatkan keamanan jaringan. Laporan ini berfungsi sebagai alat penting bagi organisasi untuk memahami sifat serangan dan menerapkan langkah-langkah keamanan yang diperlukan serta memberikan cara mencegah serangan di masa mendatang.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil Pengumpulan Data

Pada tahap ini, berhasil dikumpulkan berbagai jenis *log* dari *web server* PT Herbal Pharmaceutical yang mengalami insiden keamanan. Data yang dikumpulkan meliputi *Access Log* mencatat semua akses ke sistem, *Error Log* mencatat kesalahan yang terjadi di sistem, *Auth Log* mencatat aktivitas login atau otentikasi dan *Syslog* mencatat aktivitas umum sistem operasi. Seperti pada tabel 4.1 meliputi:

Tabel 4.1 Jenis Data *Log* yang diproses

Jenis Log	Jumlah Events	Rentang Waktu	Size (MB)
Access Log	3,247,384 events	30 hari	63.7 MB
Error Log	774 events	30 hari	154.9 KB
Auth Log	183 events	30 hari	18.4 KB
Syslog	23,891 events	30 hari	511.7 KB

Pada tabel 4.1 bahwa *access log* mencatat semua akses atau interaksi pengguna dengan sistem *web server* dengan total *events* ada 3.247.384 *event* kejadian yang tercatat. Ini menunjukkan bahwa ada sangat banyak aktivitas yang terjadi di *web server* tersebut. Data ini dikumpulkan selama 30 hari. Ukuran *file* untuk *access log* adalah 63.7 MB. Ukuran ini relatif besar karena mencatat setiap permintaan dan respons dari *web server*. *Error Log* mencatat setiap kesalahan atau kegagalan yang terjadi di dalam system, terdapat 774 *event* kesalahan. Data ini juga dikumpulkan selama 30 hari. Ukuran *file* untuk *error log* adalah 154.9 KB. Ukurannya jauh lebih kecil dibandingkan *access log* karena hanya mencatat kejadian error. *Auth Log* mencatat semua aktivitas login atau proses otentikasi lainnya yang terjadi di sistem. Ada 183 *event* yang berkaitan dengan otentikasi. Data ini juga dikumpulkan selama 30 hari. Ukuran *file* untuk *auth log* adalah 18.4 KB, yang paling kecil di antara *log* lainnya. *Syslog* merekam berbagai aktivitas umum yang terjadi pada sistem operasi. Ada 23.891 *event* yang tercatat. Dikumpulkan selama 30 hari serta memiliki ukuran *file* 511.7 KB. Secara

keseluruhan, tabel ini memberikan gambaran umum tentang jenis dan volume data *log* yang diproses dalam penelitian ini, di mana *Access Log* memiliki jumlah *event* dan ukuran *file* terbesar, menunjukkan banyaknya interaksi yang terjadi pada *web server*. Data ini kemudian digunakan untuk mengidentifikasi pola-pola mencurigakan dalam proses analisis forensik digital.

Proses pengumpulan dimulai dari data dikumpulkan menggunakan protokol *SSH* (*Secure Shell*) merupakan protokol jaringan kriptografi yang digunakan untuk mengakses dan mengelola perangkat secara aman melalui jaringan internet dan *SCP (Secure Copy Protocol)* digunakan untuk menyalin *file log* dari server ke tempat penyimpanan dengan aman, memastikan tidak ada data yang rusak atau diubah selama transfer untuk memastikan integritas data selama proses transfer. Setiap *file log* diverifikasi menggunakan *checksum MD5* untuk memastikan tidak ada perubahan atau korupsi data selama proses pengumpulan sehingga hasil analisisnya bisa dipercaya.

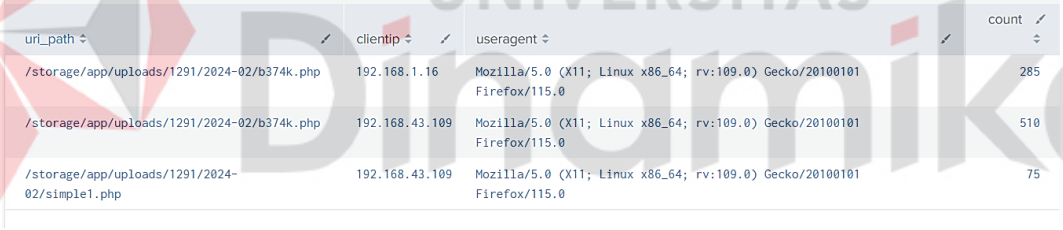
4.2 Hasil Pengolahan Data

Pada tahapan ini dilakukan pengolahan data setelah data dikumpulkan untuk mendeteksi, menganalisis, dan melaporkan adanya serangan *Backdoor Shell* dan *Broken Access Control*, yang terjadi pada web server PT Herbal Pharmaceutical. Berikut adalah detail tujuan dari setiap sub-bagian:

4.2.1 Hasil *Searching Backdoor Shell*

Pada bagian ini, akan membahas secara rinci mengenai deteksi dan analisis serangan *backdoor shell* yang terjadi. Serangan ini teridentifikasi melalui aktivitas mencurigakan berupa pengunggahan *file b374k.php* ke direktori */storage/app/uploads*/1291/2024-02/. *File b374k.php* ini diidentifikasi sebagai *backdoor shell*, yaitu sebuah program berbahaya yang memungkinkan penyerang mendapatkan akses dan kontrol penuh terhadap sistem secara diam-diam. Alamat IP 192.168.43.109 terdeteksi sebagai sumber utama serangan ini. IP tersebut tercatat melakukan 374 aktivitas berbahaya menggunakan *backdoor shell* tersebut. Tingginya frekuensi aktivitas dari IP ini mengindikasikan adanya upaya berkelanjutan untuk mengeksploitasi sistem.

Serangan *Backdoor Shell* sering menggunakan metode *POST* untuk mengunggah *file* berbahaya atau mengirimkan perintah ke server karena metode ini dapat membawa lebih banyak data dalam badan permintaan dibandingkan metode *GET*. | *stats count by uri_path, clientip, useragent* perintah *transforming command* yang menghitung dan mengelompokkan data. *Stats count* untuk menghitung jumlah kejadian (*event*) yang cocok dengan kriteria sebelumnya. *By uri_path, clientip, useragent* untuk mengelompokkan hasil hitungan berdasarkan kombinasi unik dari jalur *URI (uri_path*), alamat *IP* klien (*clientip*), dan *user agent*. Ini membantu mengidentifikasi *file* mana yang diakses, dari *IP* mana, dan menggunakan perangkat atau *browser* apa, serta berapa kali kombinasi tersebut terjadi. | *where count > 5* untuk menampilkan hasil di mana jumlah kejadian (*count*) seperti kombinasi *uri_path, clientip*, dan *useragent* lebih dari 5. Tujuannya adalah untuk mengurangi *noise* dan fokus pada aktivitas yang berulang atau mencurigakan, karena serangan *backdoor shell* sering melibatkan banyak akses berulang ke *file* yang diunggah. Seperti pada gambar 4.1 dibawah ini:



uri_path	clientip	useragent	count
/storage/app/uploads/1291/2024-02/b374k.php	192.168.1.16	Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0	285
/storage/app/uploads/1291/2024-02/b374k.php	192.168.43.109	Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0	510
/storage/app/uploads/1291/2024-02/simple1.php	192.168.43.109	Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0	75

Gambar 4.1 Indikasi *Backdoor Shell*

Pada gambar 4.1 dijelaskan bahwa kolom *uri_path* menunjukkan lokasi atau alamat *file* yang diakses di server. Kolom *clientip* berisi alamat *Internet Protocol* (*IP)* dari komputer atau perangkat yang melakukan akses ke *file* tersebut. Setiap alamat *IP* unik untuk setiap perangkat yang terhubung ke internet. Kolom *useragent* memberikan informasi tentang perangkat lunak seperti *browser* web yang digunakan oleh pengguna untuk mengakses *file* tersebut. Kolom *count* menunjukkan berapa kali *file* di *uri_path* tertentu diakses dari *clientip* dan *useragent* yang spesifik. Ini adalah jumlah total akses. Dapat disimpulkan bahwa ada dua *file* yang berbeda yang diakses yaitu *b374k.php* dan *simple1.php*. *File-file* tersebut diunggah ke server pada bulan Februari 2024. Akses dilakukan dari dua alamat *IP* internal yang berbeda yaitu *192.168.1.16* dan *192.168.43.109.* Semua

akses yang tercatat dalam tabel ini dilakukan menggunakan browser *Firefox* versi 115.0 di sistem operasi *Linux. File b374k.php* diakses lebih sering secara total keseluruhan 795 kali dibandingkan dengan *simple1.php*. Alamat *IP 192.168.43.109* lebih aktif dalam mengakses *file-file* tersebut dibandingkan *IP 192.168.1.16*.

4.2.2 Hasil *Searching Broken Access Control*

Pada bagian ini, dilakukan penguraian temuan penting terkait upaya peretasan atau penyalahgunaan yang terdeteksi pada panel administrasi web server. Insiden ini teridentifikasi melalui pemantauan log aktivitas penyimpanan perubahan data evaluasi. Adanya serangan *Broken Access Control*, yaitu kondisi di mana pengguna dapat mengakses atau melakukan tindakan yang seharusnya tidak diizinkan, karena sistem tidak memeriksa hak akses dengan benar. Tanda minus (-) menunjukkan sistem tidak mengautentikasi pengguna untuk jenis permintaan ini. Seperti pada gambar 4.2 dibawah ini:

clientip	user	url_path	count
192.168.43.109	-	/admin/evaluasi/edit-save/1	45
192.168.43.109	-	/admin/evaluasi/edit-save/2	45
192.168.43.109	-	/admin/evaluasi/edit-save/10	15
192.168.43.109	-	/admin/evaluasi/edit-save/11	15
192.168.43.109	-	/admin/evaluasi/edit-save/12	15
192.168.43.109	-	/admin/evaluasi/edit-save/13	15
192.168.43.109	-	/admin/evaluasi/edit-save/14	15
192.168.43.109	-	/admin/evaluasi/edit-save/15	15
192.168.43.109	-	/admin/evaluasi/edit-save/16	15
192.168.43.109	-	/admin/evaluasi/edit-save/3	15
192.168.43.109	-	/admin/evaluasi/edit-save/4	15
192.168.43.109	-	/admin/evaluasi/edit-save/5	15
192.168.43.109	-	/admin/evaluasi/edit-save/6	15
192.168.43.109	-	/admin/evaluasi/edit-save/7	15
192.168.43.109	-	/admin/evaluasi/edit-save/8	15
192.168.43.109	-	/admin/evaluasi/edit-save/9	15

Gambar 4.2 Indikasi *Broken Access Control*

Pada gambar 4.2 menampilkan sebuah tabel data *log* atau catatan aktivitas dari sebuah sistem seperti *clientip* untuk menunjukkan alamat *Internet Protocol* (*IP)* dari klien atau perangkat yang melakukan akses. Dalam gambar ini, semua entri menunjukkan alamat *IP* yang sama yaitu 192.168.43.109. Ini adalah alamat *IP* pribadi (*private IP address*) yang sering digunakan di jaringan lokal, ini mengindikasikan bahwa semua aktivitas ini berasal dari satu perangkat dalam jaringan lokal tersebut. Kolom *user* untuk menampilkan nama pengguna (*username*) yang melakukan aktivitas. Namun, dalam semua baris, kolom ini kosong ("-"). *Url_path* untuk menunjukkan jalur *Uniform Resource Locator* (*URL*)

yang diakses oleh klien yang merupakan bagian dari alamat *web* yang mengidentifikasi sumber daya spesifik di server. Kolom *count* untuk menunjukkan berapa kali sebuah "*url_path*" tertentu diakses oleh "*clientip*" yang sama.

4.2.3 Hasil Data *Filtering Backdoor Shell*

Pada tahap ini adanya penerapan *filtering* untuk memilih subset data berdasarkan kriteria tertentu. Kode status *HTTP* 4xx (*Client Error*) mengindikasikan adanya masalah seperti kode 400-an seperti, 401 *Unauthorized*, 403 *Forbidden*, 404 *Not Found* menunjukkan bahwa ada masalah dengan permintaan dari sisi klien. Ini sering kali merupakan indikasi aktivitas mencurigakan seperti upaya *brute force*, pemindaian *bot*, atau pencarian celah keamanan. Dengan memfilter status>=400, kueri ini secara efektif hanya akan melihat *event* yang menunjukkan adanya kesalahan atau masalah. | *stats count by status, uri_path* perintah *stats* digunakan untuk melakukan agregasi statistik. *Count* untuk menghitung berapa kali setiap kombinasi *status* dan `*uri_path*` muncul dalam data yang telah di filter. *By status, uri_path* untuk mengelompokkan hasil hitungan berdasarkan dua *field* (kolom) ini. Artinya, *Splunk* akan menghitung jumlah kemunculan unik untuk setiap pasangan kode status *HTTP* seperti, 404 dan jalur *URI* seperti, /*file*-yang-tidak-ada.*php*. | *sort -count* perintah *sort* untuk mengurutkan hasil. -*count* untuk mengurutkan hasil berdasarkan kolom *count* (jumlah) secara menurun dari yang terbesar ke terkecil. Ini sangat berguna karena akan menampilkan kesalahan yang paling sering terjadi di bagian atas hasil, sehingga memudahkan identifikasi masalah paling dominan atau serangan yang paling aktif. Hasilnya akan diurutkan dari yang paling sering terjadi ke yang paling jarang. Pencarian ini telah telah menghasilkan 10.000 hasil dengan memindai 4.465.153 *events* dalam 62,684 detik. Seperti pada gambar 4.3 berikut:

status	uri_path	count
404	/vendor/crudbooster/assets/adminlte/plugins/jQuery/jquery-2.2.3.min.js	5764
500	/uploads/1289/2024-05/image1.jpg	187
500	/	121
403	/.hta	110
403	/.hta_	110
403	/.htaccess	110
403	/.htaccess_	110
403	/.htpasswd	110
403	/.htpasswd_	110
404	/.bash_history	110

Gambar 4.3 Hasil *Filtering Backdoor Shell*

Pada gambar 4.3 berisi beberapa status *HTTP error, path URI*, dan jumlah kejadian (*count*) dari *error* tersebut. Di awal tabel, dapat dilihat beberapa *error 404* dan *500*. *Error* 404 (*Not Found*) terjadi saat server tidak bisa menemukan *file* yang diminta, seperti *jquery-2.2.3.min.js* yang hilang dan terjadi sebanyak 5764 kali. Ini bisa berarti ada *file* yang salah lokasi atau memang sudah tidak ada. Kemudian, ada error 500 (*Internal Server Error*) yang menunjukkan ada masalah di sisi server, misalnya pada *file image1.jpg* atau akses langsung ke *root* (*/*), yang masing-masing terjadi 187 dan 121 kali. *Error 500* ini perlu segera diperiksa karena berarti ada yang salah dengan kode program atau konfigurasi server itu sendiri. Sebagian besar baris menampilkan *error* 403 (*Forbidden*) yang terjadi untuk berbagai *file* yang terkait dengan konfigurasi atau otentikasi, seperti *.hta, .htaccess, .htpasswd,* dan variannya dengan *underscore (_)*. Masing-masing *error* ini terjadi 110 kali. Status 403 berarti server menolak akses ke *file-file* tersebut, yang sebenarnya adalah hal yang baik dari sisi keamanan karena *file-file* tersebut berisi informasi sensitif dan tidak boleh diakses langsung dari *web*. Terakhir, ada juga *error 404* untuk *.bash_history* sebanyak 110 kali, yang menunjukkan server tidak menemukan *file* riwayat perintah *bash*.

4.2.4 Hasil Data Filtering Broken Access Control

Pola ini dirancang untuk mencari upaya akses ke *file* konfigurasi sensitif yang umum pada *web server apache*, seperti *.htaccess* atau *.htpasswd*. *File-file* ini berisi informasi penting dan tidak boleh diakses secara publik. *Status=404 OR status=403* untuk memfilter *event* berdasarkan kode status *HTTP* yang dihasilkan. Status=404 menunjukkan bahwa *file* atau sumber daya yang diminta tidak

ditemukan di *server*. Jika penyerang mencoba mencari *file* atau direktori sensitif yang tidak ada, ini akan menghasilkan kode 404. Status=403 menunjukkan bahwa *server* menolak akses ke sumber daya yang diminta, meskipun *file* tersebut mungkin ada. Ini adalah tanda positif dari sisi keamanan jika *file* sensitif berhasil diblokir, tetapi juga bisa mengindikasikan upaya penyerang untuk melewati kontrol akses. | *stats count by uri_path, clientip* dengan operator pipa (|) meneruskan hasil dari perintah sebelumnya ke perintah berikutnya. Perintah *stats* menghitung jumlah kemunculan (*count*) untuk setiap kombinasi unik dari *uri_path* (jalur *URI* yang diakses) dan *clientip* (alamat *IP* klien yang melakukan permintaan).

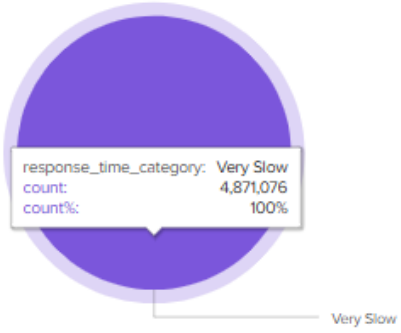
uri_path	clientip	count
/.bash_history	10.13.17.17	88
/.bash_history	192.168.43.109	22
/.bashrc	10.13.17.17	88
/.bashrc	192.168.43.109	22
/.cache	10.13.17.17	88
/.cache	192.168.43.109	22
/.config	10.13.17.17	88
/.config	192.168.43.109	22
/.cvs	10.13.17.17	88
/.cvs	192.168.43.109	22

Gambar 4.4 Hasil *Filtering Broken Access Control*

Gambar 4.4 menampilkan tabel yang mencatat upaya akses ke berbagai *file* dari direktori sensitif di sebuah sistem, lengkap dengan alamat *IP* klien yang mencoba mengaksesnya dan berapa kali percobaan akses itu terjadi. Ini seperti daftar alarm keamanan yang berbunyi ketika ada yang mencoba melihat atau mengakses *file-file* penting. Pencarian ini telah menghasilkan 698 hasil dengan memindai 10,640 *events* dalam 74,203 detik. Terdapat dua alamat *IP* yaitu *IP 10.13.17.17* dan *IP 192.168.43.109* yang berulang kali mencoba mengakses *file-file* seperti `.*bash_history*`, `.*bashrc*` (*file* konfigurasi *shell*), `.*cache*` (folder *cache*), `.*config*` (folder konfigurasi), dan `.*cvs*` . *IP 10.13.17.17* selalu memiliki jumlah percobaan yang lebih tinggi sebanyak 88 kali dibandingkan *IP 192.168.43.109* sebanyak 22 kali untuk setiap *file* yang sama. Pola ini menunjukkan adanya upaya untuk mencari dan mungkin mengeksploitasi informasi sensitif di sistem oleh kedua *IP* tersebut.

4.2.5 Hasil Data *Modification*

Pada tahap ini menganalisis data *log web server* dengan cara menghitung *response_time* dan mengategorikannya menjadi empat kelompok yaitu "*Fast*" (kurang dari 100), "Normal" (antara 100 dan 499), "*Slow*" (antara 500 dan 999), dan "*Very Slow*" (1000 atau lebih). Setelah mengategorikan waktu respons, *query* ini kemudian menghitung jumlah kejadian (*count*) untuk setiap kategori waktu *respons* (*response_time_category*). Ini akan mengevaluasi kondisi secara berurutan dan menetapkan kategori berdasarkan kondisi pertama yang terpenuhi seperti *response_time<100, "Fast"* jika waktu *respons (response_time)* kurang dari 100, maka *response_time_category* akan diatur sebagai *"Fast"*. *Response_time<500,* "Normal" jika waktu respons kurang dari 500 tetapi tidak kurang dari 100, maka kategorinya adalah "Normal". *Response_time<1000, "Slow"* jika waktu respons kurang dari 1000 tetapi tidak kurang dari 500, maka kategorinya adalah *"Slow"*. 1=1, *"Very Slow"* ini adalah kondisi "*catch-all*" atau *default*. Jika tidak ada kondisi sebelumnya yang terpenuhi yaitu, waktu respons 1000 atau lebih, maka kategorinya adalah "*Very Slow*". Angka 1=1 selalu benar, sehingga ini akan menangkap semua nilai yang tersisa. Tujuan dari bagian ini adalah untuk mengubah nilai numerik waktu respons menjadi kategori yang lebih mudah dipahami dan dianalisis, memberikan gambaran kualitatif tentang kinerja server. | *stats count by response_time_category.* Perintah *stats* digunakan untuk melakukan agregasi statistik. Perintah *count* untuk menghitung jumlah *event* (permintaan) yang termasuk dalam setiap kategori. *By response_time_category* untuk mengelompokkan hasil hitungan berdasarkan *field response_time_category* yang baru saja dibuat. Ini akan menghasilkan tabel yang menunjukkan berapa banyak permintaan yang "*Fast*", "Normal", "*Slow*", dan "*Very Slow"*. Seperti pada gambar 4.5 berikut:



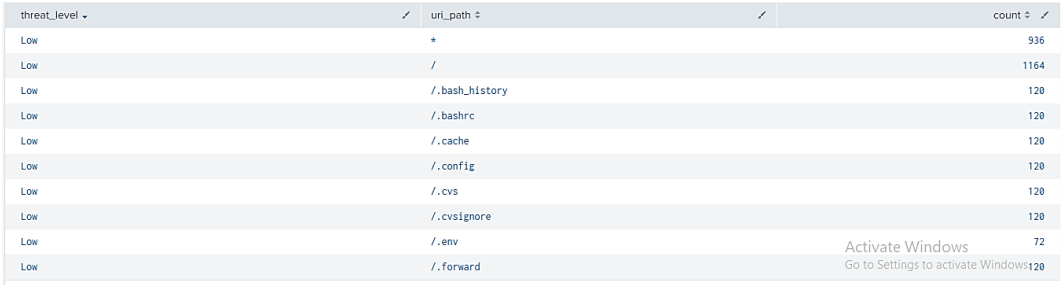
Gambar 4.5 Hasil *Response Time*

Gambar 4.5 menampilkan sebuah diagram lingkaran yang menunjukkan kategori waktu respons dan jumlahnya. Terlihat jelas bahwa seluruh data, yaitu 100%, masuk ke dalam kategori "*Very Slow*" dengan total 4.871.076 entri. Ini mengindikasikan bahwa semua respons yang diukur dalam data ini memiliki kinerja yang sangat lambat.

4.2.6 Hasil Data *Insertion Backdoor Shell*

Pada bagian ini menjelaskan bagaimana mengidentifikasi potensi aktivitas berbahaya pada web server dengan menganalisis informasi *User-Agent* dari setiap akses. *Useragent* merupakan *field* yang berisi informasi tentang perangkat lunak seperti web browser, *bot*, skrip yang digunakan klien untuk mengakses server. .*(*curl*|*wget*|*python*).* adalah pola ekspresi reguler yang mencari salah satu dari kata kunci "*curl*", "*wget*", atau "*python*". Alat-alat ini *curl, wget* atau bahasa pemrograman *python* sering digunakan oleh *bot*, skrip otomatis, atau penyerang untuk melakukan pemindaian, unduhan, atau interaksi otomatis dengan *web* server, yang bisa mengindikasikan aktivitas berbahaya. Status *"High", "Low"* jika *useragent* cocok dengan pola yang dicari mengandung "*curl", "wget*", atau "*python"*, *threat_level* akan diatur sebagai "*High*". Jika tidak cocok, *threat_level* akan diatur sebagai "*Low*".*By threat_level, uri_path.*

Pencarian ini menghasilkan 272.734 hasil dengan memindai 4.871.076 *events* dalam 109,783 detik. Untuk mengidentifikasi potensi ancaman berdasarkan *useragent* dan kemudian menghitung seberapa sering ancaman tersebut muncul untuk setiap jalur *URI (uri_path*). Secara detail, bagian *eval* akan membuat sebuah bidang baru bernama *threat_level*. Tingkat ancaman ini akan di set menjadi "*High*" jika *useragent* yang mengakses server mengandung kata "*curl", "wget*", atau "*python*" yang seringkali digunakan oleh *bot* atau skrip otomatis. Jika *useragent* tidak mengandung salah satu dari kata kunci tersebut, *threat_level* akan diset menjadi "*Low*". Setelah itu, bagian *stats* akan menghitung jumlah kemunculan (*count*) dari setiap kombinasi *threat_level* dan *uri_path*, sehingga dapat melihat jalur mana yang paling sering diakses oleh potensi ancaman ("*High" threat level*) atau oleh pengguna biasa ("*Low" threat level*).



threat_level	uri_path	count
Low	*	936
Low	/	1164
Low	/.bash_history	120
Low	/.bashrc	120
Low	/.cache	120
Low	/.config	120
Low	/.cvs	120
Low	/.cvsignore	120
Low	/.env	72
Low	/.forward	120

Gambar 4.6 Hasil *Useragent Backdoor Shell*

Dari gambar 4.6, bahwa semua entri yang terlihat memiliki *threat_level "Low"*. Ini berarti semua aktivitas yang terekam tidak terdeteksi sebagai ancaman tinggi (misalnya, tidak ada akses dari bot atau skrip mencurigakan seperti *curl* atau *wget*). Kolom *uri_path* menunjukkan berbagai jalur yang diakses, seperti */, *, /bash_history, /cache*, dan lain-lain. Kolom *Count 1* menunjukkan berapa kali jalur tersebut diakses. Misalnya, jalur * dan / memiliki jumlah akses yang paling tinggi, yaitu 780 dan 670, sementara banyak jalur lainnya seperti */bash_history* atau */cache* diakses sebanyak 100 kali. Singkatnya, tabel ini merangkum akses ke berbagai bagian server, dan semuanya dianggap memiliki tingkat ancaman rendah.

4.2.7 Hasil Data *Insertion Broken Access Control*

Pada bagian ini, menjelaskan bagaimana menganalisis *log* akses web server untuk mendeteksi upaya akses ke lokasi atau jalur yang dianggap sensitif. Tujuannya adalah untuk mengidentifikasi potensi serangan *Broken Access Control*, di mana pengguna mungkin mencoba mengakses bagian website yang seharusnya tidak bisa mereka jangkau.

Akses tidak sah ke jalur ini dapat mengindikasikan serangan *broken access control*. Kolom akan bernilai "*Yes*" jika jalur *URI* yang diakses (*uri_path*) adalah */admin/* atau */config/* yang seringkali merupakan lokasi sensitif atau administratif. Jika bukan kedua jalur tersebut, *is_sensitive* akan bernilai "*No*". Setelah itu, perintah *stats* akan menghitung jumlah akses (*count*) untuk setiap kombinasi *is_sensitive* dan alamat *IP* klien (*clientip*). Sehingga, *query* ini akan menunjukkan berapa banyak kali setiap alamat *IP* mengakses jalur sensitif ("*Yes*") dan berapa banyak kali mereka mengakses jalur non-sensitif ("*No*"). Ini sangat berguna untuk mengidentifikasi

siapa saja yang sering mengakses bagian-bagian penting dari *website*. Seperti pada gambar 4.7 berikut:

is_sensitive	clientip
No	10.13.1.83
No	10.13.17.17
No	172.20.10.4
No	192.168.0.129
No	192.168.1.10
No	192.168.1.16
No	192.168.100.52
No	192.168.211.28
No	192.168.43.109
No	192.168.43.28
No	::1
No	\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
Yes	10.13.17.17
Yes	192.168.43.109

Gambar 4.7 Hasil *Insertion Broken Access Control*

Dari gambar 4.7 terlihat banyak baris di mana *is_sensitive* bernilai *"No",* diikuti oleh berbagai alamat *IP* seperti *10.13.1.83, 192.168.0.129,* dan sebagainya. Ini berarti banyak akses tercatat ke jalur-jalur yang tidak dianggap sensitif. Namun, di bagian bawah tabel yang terlihat, ada juga beberapa baris di mana *is_sensitive* bernilai "*Yes*", seperti untuk *clientip 10.13.17.17* dan *192.168.43.109*. Ini menandakan bahwa alamat-alamat *IP* tersebut telah mengakses setidaknya satu jalur yang dikategorikan sebagai sensitif *(/admin/ atau /config/).*

4.2.8 Hasil Data *Deletion Backdoor Shell*

Pada bagian ini, menjelaskan bagaimana melakukan analisis terhadap *log* akses web server dengan fokus pada permintaan yang mengarah ke sumber daya dinamis (halaman web interaktif, *script*, atau data) dan mengabaikan permintaan untuk aset statis (seperti gambar atau *stylesheet*). Tujuan dari pendekatan ini adalah untuk mendapatkan gambaran yang lebih jelas tentang bagaimana pengguna atau script berinteraksi dengan fungsionalitas utama website, bukan hanya elemen visualnya.

method	uri_path	count
GET	/	96
GET	/.bash_history	24
GET	/.bashrc	24
GET	/.cache	24
GET	/.config	24
GET	/.cvs	24
GET	/.cvsignore	24
GET	/.env	24
GET	/.forward	24
GET	/.git/HEAD	24

Gambar 4.8 Hasil Data *Delete Backdoor Shell*

Pada gambar 4.8 hanya menampilkan sebagian dari tabel hasil analisis *log web server* yang telah disaring, tidak termasuk permintaan untuk *file CSS, JS*, atau *PNG*. Tabel ini menunjukkan tiga kolom: `*method*` (metode *HTTP* yang digunakan), `*uri_path*` (jalur atau halaman yang diakses), dan `*count*` (jumlah kali akses). Dari data yang terlihat, semua entri menggunakan metode *HTTP "GET*". Ini berarti semua permintaan ini adalah upaya untuk mengambil atau membaca data dari *server*. Kolom `*uri_path*` menunjukkan berbagai jalur yang diakses, seperti `/` halaman utama. */bash_history, ./bashrc, ./cache*, dan lain-lain. Kolom *count*` menunjukkan berapa kali jalur tersebut diakses dengan metode *GET*. Jalur `/` halaman utama diakses sebanyak 96 kali, yang paling banyak di antara yang terlihat. Sementara itu, jalur-jalur lain seperti *`./bash_history`, `./bashrc`, `./cache`,* dan lainnya diakses masing-masing sebanyak 24 kali.

4.2.9 Hasil Data *Deletion Broken Access Control*

Pada bagian ini, untuk menganalisis *log* akses web server dengan tujuan mengidentifikasi aktivitas yang tidak normal atau berpotensi tidak sah pada akses yang tidak sesuai dengan pola normal, terutama yang berkaitan dengan akses ke area administratif dengan cara menyaring lalu lintas yang sah dan menyoroti kejadian yang memerlukan investigasi lebih lanjut.

Semua *event* dengan status=200 tetapi *user_role* bukan "*admin*" sehingga ini bisa mengindikasikan akses berhasil oleh pengguna *non-admin* ke area yang seharusnya terbatas, atau aktivitas lain yang perlu diperhatikan. Tujuan dari bagian ini adalah untuk menghilangkan lalu lintas normal yang dilakukan oleh administrator yang sah, sehingga lebih mudah untuk menemukan anomali atau aktivitas mencurigakan.

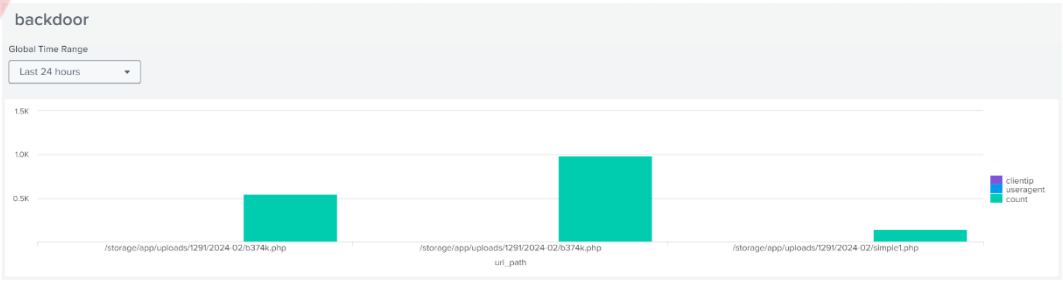
uri_path	status	count
*	200	936
/	200	1008
/	304	24
/	500	132
/.bash_history	404	120
/.bashrc	404	120
/.cache	404	120
/.config	404	120
/.cvs	404	120
/.cvsignore	404	120

Gambar 4.9 Hasil Data *Delete Broken Access Control*

Pada gambar 4.9 menampilkan sebagian dari tabel hasil analisis *log web server* yang telah disaring, berfokus pada akses yang bukan "*status 200* dari *admin*". Tabel ini memiliki tiga kolom: *uri_path* merupakan jalur atau halaman yang diakses, kode status *HTTP* yang diterima, dan *count* (jumlah kejadian). Dari tabel, terdapat berbagai jalur *URI* dan status yang terkait dengan jalur * mendapatkan status 200 (berhasil) sebanyak 936 kali. Jalur atau (halaman utama) mendapatkan status 200 sebanyak 1008 kali, status 304 (tidak dimodifikasi, menggunakan *cache*) sebanyak 24 kali, dan status 500 (*server error*) sebanyak 132 kali. Kehadiran status 500 untuk jalur utama ini menunjukkan ada masalah serius di sisi server. Banyak jalur lainnya seperti *./bash_history, ./bashrc, ./cache*, dan lain-lain, semuanya mendapatkan status 404 (tidak ditemukan) sebanyak 120 kali. Ini bisa mengindikasikan upaya untuk mengakses file atau direktori yang tidak ada, atau adanya aktivitas pemindaian.

4.3 Hasil Analysis

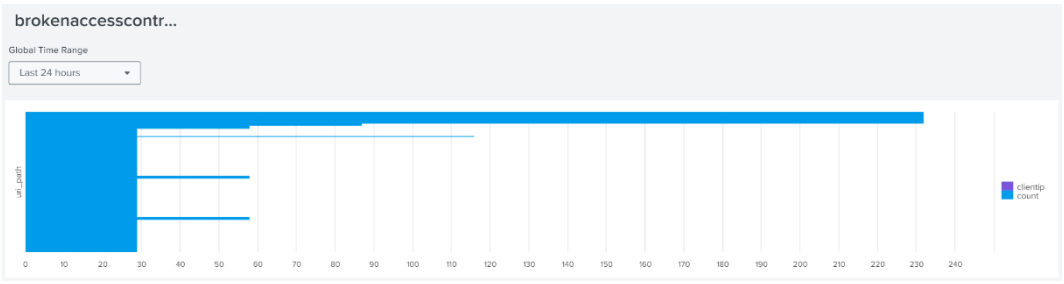
Pada tahap analisis ini, setelah melakukan identifikasi pola-pola spesifik yang menunjukkan adanya serangan *Backdoor Shell* dan *Broken Access Control* berdasarkan *data log* yang telah dikumpulkan dan diperiksa. Berikut adalah rincian temuan penting:



Gambar 4.10 Grafik *Backdoor Shell*

Pada gambar 4.10 dari hasil analisis, *file /storage/app/uploads/*1291/2024/02 */b374k.php* diakses sekitar 500 kali.*File /storage/app/uploads/*1291/2024/02/ *b374k.php* merupakan yang paling sering diakses, dengan jumlah mendekati 1000 kali. *File /storage/app/uploads/*1291/2024/02/*simple1.php* memiliki akses paling sedikit, yaitu sekitar 100-200 kali. Secara keseluruhan, analisis grafik ini menyoroti pola akses yang tidak biasa terhadap *file-file* tertentu di direktori unggahan, dengan

satu *file b374k.php* menunjukkan tingkat akses yang jauh lebih tinggi dalam periode 24 jam terakhir.



Gambar 4.11 Grafik *Broken Access Control*

Pada gambar 4.11 dijelaskan bahwa analisis menunjukkan adanya satu jalur *URL* spesifik (ditunjukkan oleh batang paling atas) yang menjadi target insiden atau percobaan "*broken access control*" paling dominan. Jalur ini mencatat sekitar 230 kejadian dalam sehari, mengindikasikan upaya akses yang signifikan. Dua jalur *URL* lainnya (batang kedua dan ketiga dari atas) juga tercatat sebagai target, masing-masing dengan jumlah kejadian sekitar 60 kali. Ini menunjukkan adanya upaya serupa namun dengan frekuensi yang lebih rendah dibandingkan target utama. Satu jalur *URL* (batang paling bawah) menunjukkan aktivitas paling rendah, dengan sekitar 30 kejadian dalam periode yang sama.

Gambar 4.12 Kode *Forbidden*

Gambar 4.12 merupakan *log* sistem yang mengindikasikan serangkaian kesalahan *HTTP* status *code 403 Forbidden*. Kesalahan ini menunjukkan bahwa server web telah menerima dan memahami permintaan dari klien, namun menolak untuk memenuhinya karena kurangnya otorisasi atau hak akses. Status kolom ini secara konsisten menampilkan kode 403, yang mengonfirmasi bahwa setiap upaya akses ke URI (Uniform Resource Identifier) yang terdaftar telah ditolak oleh server.

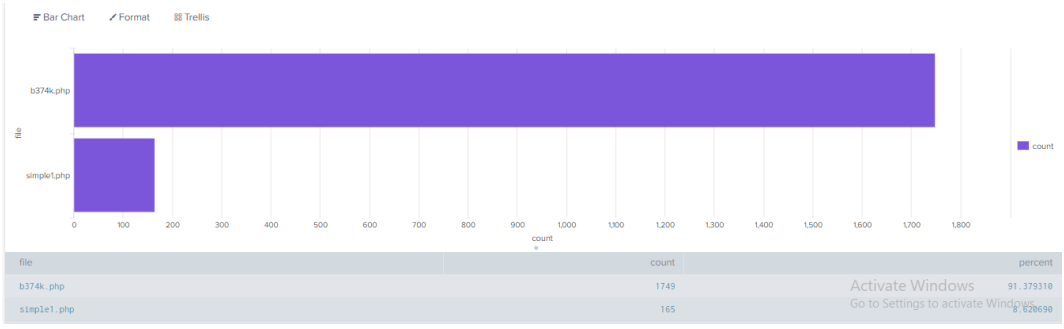
Kolom *URI_Path* ini mencantumkan jalur spesifik dari sumber daya yang berusaha diakses. Jalur-jalur yang terlihat, seperti */ .hta, / .htaccess, / .htpasswd*, merujuk pada *file-file* konfigurasi krusial pada server web berbasis *Apache. File .htaccess* dan *.htpasswd* mengandung pengaturan direktori dan kredensial autentikasi, yang mana akses langsungnya oleh publik secara sengaja dibatasi untuk menjaga integritas dan keamanan sistem. Selain itu, terdapat juga entri */app/,* yang kemungkinan besar merujuk pada direktori aplikasi atau folder lain yang tidak diizinkan untuk diakses secara langsung melalui antarmuka web. Kolom *count* ini menunjukkan frekuensi terjadinya kesalahan 403 untuk setiap jalur *URI.* Angka 200 pada sebagian besar entri, dan 112 untuk */app/,* mengindikasikan adanya sejumlah besar upaya akses yang tidak diotorisasi ke sumber daya-sumber daya tersebut.

4.4 Hasil Evaluasi

Tahap pelaporan ini adalah langkah akhir dalam penelitian, di mana semua temuan, analisis, dan kesimpulan disajikan secara komprehensif dan mudah dimengerti. Laporan ini disusun untuk memberikan gambaran yang jelas mengenai insiden keamanan siber yang terjadi pada *web server* PT Herbal Pharmaceutical dan rekomendasi untuk peningkatan keamanan di masa mendatang.

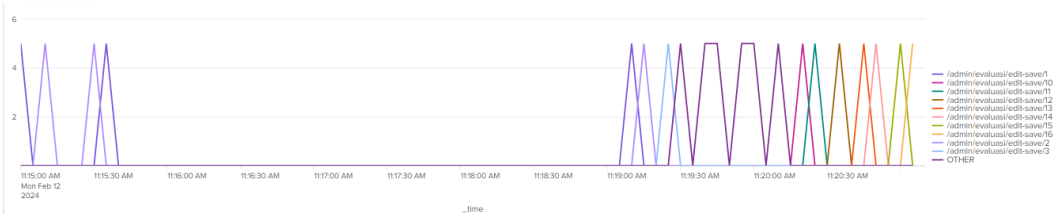
4.4.1 Hasil Evaluasi Deteksi *Backdoor Shell* dan *Broken Access Control*

Pada tahap ini dijelaskan bahwa hasil dari langkah-langkah setiap proses yang sudah dilakukan dan memberikan hasil bahwa PT Herbal Pharmaceutical terindikasi penyerangan Backdoor Shell dan Broken Access Control seperti pada gambar 4.13 dan gambar 4.14:



Gambar 4.13 *Backdoor Shell*

Berdasarkan gambar 4.13, dapat disimpulkan bahwa dari total hitungan (1749 + 165 = 1914), *file* bernama *b374k.php* memiliki jumlah yang jauh lebih banyak yaitu 1749 atau sekitar 91.38% dari total. Sementara itu, *file simple1.php* hanya memiliki jumlah sebanyak 165 atau sekitar 8.62% dari total. Visualisasi *bar chart* memberikan perbandingan yang intuitif, sementara tabel data menyediakan angka-angka yang presisi untuk mendukung perbandingan tersebut.



Gambar 4.14 *Broken Access Control*

Pada gambar 4.14 merupakan visualisasi yang sering digunakan untuk mendeteksi serangan *Broken Access Control* atau aktivitas mencurigakan lainnya di mana penyerang mencoba mengakses berbagai sumber daya dalam hal ini, ID evaluasi 1 hingga 16 secara berurutan. Pola akses yang melonjak, berhenti, lalu melonjak lagi dengan cepat mencoba banyak ID berbeda dalam rentang waktu singkat sekitar 5 menit sangat tidak biasa untuk perilaku pengguna normal dan dapat mengindikasikan serangan otomatis atau skrip yang mengeksploitasi celah keamanan.

4.4.2 Hasil Evaluasi Efektivitas Kinerja *SPL*

Pada tahap ini menganalisis data *log* menggunakan platform seperti Splunk, efektivitas kinerja pencarian memegang peranan krusial untuk memastikan proses eksplorasi dan identifikasi anomali dapat berjalan secara optimal. Efektivitas kinerja pencarian merujuk pada kecepatan dan efisiensi platform dalam memproses kueri dan menghasilkan data yang relevan dari volume *log* yang besar. Ini mencakup durasi yang dibutuhkan untuk memindai jutaan event, mencocokkan kriteria pencarian, dan mengkompilasi hasil akhir. Mode pencarian seperti *Fast*, *Smart*, dan *Verbose* secara langsung memengaruhi aspek ini, di mana mode *Fast* dan *Smart* dirancang untuk memprioritaskan kecepatan eksekusi dengan optimasi

pemrosesan data, sementara mode *Verbose*, meskipun menampilkan detail event secara lengkap, mengorbankan kecepatan demi kelengkapan data yang diekstraksi. Seperti pada tabel 4.2:

Tabel 4.2 Efektivas Kinerja *SPL*

Fitur / Mode	Fast Mode	Smart Mode	Verbose Mode
Total Event yang Dipindai	5.276.999	5.276.999	5.276.999
Jumlah Hasil (Results)	1	1	1
Durasi Pencarian (Waktu)	0.942 detik	0.823 detik	234.434 detik (sekitar 3 menit 54 detik)
Kinerja Relatif	Cepat	Paling Cepat	Sangat Lambat
Fokus Utama	Kecepatan untuk agregasi, sedikit detail *event.*	Keseimbangan kecepatan dan kelengkapan; *default* yang cerdas.	Kelengkapan detail *event*, menampilkan semua *field.*
Kapan Digunakan	Saat butuh ringkasan cepat seperti, *count, stats, top*.	Sebagian besar waktu; untuk kebutuhan umum.	*Debugging,* eksplorasi data mendalam, atau saat butuh semua *field.*

Pada tabel 4.2 data yang disajikan dengan jelas menunjukkan perbedaan kinerja signifikan antara mode pencarian *Fast, Smart*, dan *Verbose* di *Splunk*, meskipun semuanya memindai jumlah *event* yang sama sebanyak 5.276.999 *events* dan menghasilkan jumlah hasil akhir yang identik yaitu 1 hasil. *Smart Mode* terbukti menjadi yang tercepat dengan durasi hanya 0.823 detik, diikuti ketat oleh

Fast Mode pada 0.942 detik. Kedua mode ini sangat efisien dan cocok untuk pencarian cepat atau saat dibutuhkan untuk meringkas data, dengan *Smart Mode* menjadi pilihan *default* yang menyeimbangkan kecepatan dan kelengkapan secara cerdas. Sebaliknya, *Verbose Mode* menunjukkan penurunan kinerja yang drastis, memerlukan waktu 234.434 detik, hampir 4 menit untuk menyelesaikan pencarian yang sama. Keterlambatan ekstrem ini disebabkan oleh karakteristik *Verbose Mode* yang secara agresif mengekstrak setiap *field* yang mungkin dari setiap *event,* bahkan yang tidak relevan dengan kueri. Oleh karena itu, *Verbose Mode* sebaiknya hanya digunakan untuk *debugging* mendalam atau eksplorasi data secara rinci, dan harus dihindari untuk pencarian umum guna menjaga efisiensi dan responsivitas sistem.

Setelah dilakukan proses deteksi data yang telah dikumpulkan dianalisis secara mendalam untuk mencari pola, tren, atau hubungan yang mencurigakan. Tujuan utamanya adalah menemukan bukti yang bisa menjelaskan akar penyebab terjadinya insiden. Saat menganalisis data, peneliti mencari patterns atau pola-pola tertentu, seperti aktivitas login yang tidak biasa atau perubahan file yang mencurigakan, untuk membantu mengidentifikasi apakah ada serangan yang terjadi. Seperti pada tabel 4.3:

Tabel 4.3 Ciri-ciri *Backdoor Shell* dan *Broken Access Control*

Backdoor Shell	Broken Access Control
• File dengan ekstensi *.php, .asp*, atau *.jsp* seperti b374k.php diunggah ke direktori tidak biasa ke /uploads/. 192.168.43.109 - - [02/Feb/2024:14:22:05] "POST /storage/app/uploads/b374k.php HTTP/1.1" 200 543 • Grafik 4.28 menunjukkan akses 795 kali ke *b374k.php* dalam 24 jam. • Gambar 4.21 menunjukkan status 403 berarti server menolak akses ke *file-file*	• Upaya akses ke *.htaccess, .bash_history,* atau direktori sistem menghasilkan 403 *(Forbidden).* Tabel 4.4 mencatat 110 kali percobaan akses *.htaccess* dengan status 403. • Permintaan ke */admin/, /config/,* atau *endpoint* sensitif tanpa otentikasi. 10.13.17.17 - - [05/Feb/2024:09:15:33] "GET

yang berisi informasi sensitif dan tidak boleh diakses langsung dari web	/admin/evaluasi/edit-save/1 HTTP/1.1" 302 - • Pada gambar 4.20, *Log* menunjukkan bahwa sistem tidak mengautentikasi pengguna untuk jenis permintaan ini, yang diindikasikan dengan kolom user yang kosong.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, dapat disimpulkan bahwa:

1. Penelitian ini berhasil mendeteksi dan menganalisis serangan *Backdoor Shell* dan *Broken Access Control* pada *web server* PT Herbal Pharmaceutical menggunakan platform *Splunk* dan *Search Processing Language (SPL)*. Ditemukan bukti kuat adanya upaya unggahan *file b374k.php* yang berfungsi sebagai *backdoor shell*, serta eksploitasi celah kontrol akses yang memungkinkan penyerang untuk memodifikasi dan menghapus data sensitif di area administrasi.
2. Melalui tahapan Pengumpulan, Pengolahan Data, Analisis Data, dan Evaluasi, dapat mengidentifikasi pola serangan dengan kecepatan 0.823 detik. *SPL* terbukti efektif dalam mengekstraksi informasi relevan dari *log* web server dan mengidentifikasi anomali yang menandakan aktivitas mencurigakan. Mode pencarian *Smart* di *Splunk* menunjukkan kinerja tercepat dalam memproses volume *data log* yang besar, memastikan deteksi ancaman dapat dilakukan secara optimal dan *real-time.*

5.2 Saran

Berdasarkan penelitian ini, saran yang dapat penulis berikan adalah:

1. Sangat penting untuk segera meninjau dan memperbaiki semua kelemahan dalam kontrol akses, terutama pada fitur-fitur yang memungkinkan modifikasi atau penghapusan data penting. Pastikan setiap pengguna hanya memiliki hak akses yang sangat terbatas sesuai dengan tugasnya, atau dikenal sebagai prinsip *least privilege*.
2. Adakan audit keamanan secara teratur untuk mengidentifikasi dan memperbaiki kerentanan pada web server dan aplikasi sebelum dieksploitasi oleh penyerang. Hal ini termasuk pengujian penetrasi *(pentest)* dan penilaian kerentanan secara berkala.

3. Pelatihan dan edukasi berkelanjutan kepada pengguna dan administrator mengenai praktik keamanan siber terbaik, termasuk cara mengenali serangan phishing, pentingnya kata sandi yang kuat, dan bahaya berbagi informasi sensitif.

DAFTAR PUSTAKA

A., F. (2024, September). *Web Server: Memahami Pengertian, Fungsi, dan Cara Kerjanya*. Diambil kembali dari Hostinger: https://www.hostinger.com/id/tutorial/apa-itu-web-server

Carasso, D. (2021). *Exploring Splunk.* CITO Research.

Corporation, T. M. (2015-2025). *Server Software Component: Web Shell*. Diambil kembali dari MITRE ATT&CK: https://attack.mitre.org/techniques/T1505/003/

Docs, M. W. (2025, Juli 04). *HTTP response status codes*. Diambil kembali dari HTTP Reference: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status

Elaf Almushiti, R. Z. (2024). An Investigation of Broken Access Control Types, Vulnerabilities, Protection, and Security. *Signals and communication technology*.

Gupta, N. A. (2021). Malicious Domain Detection using Data Analytics and Machine Learning Approaches. *International Journal of Network Security*.

Haitham Ameen Noman, O. M.-S. (2024). Log Poisoning Attacks in IoT: Methodologies, Evasion, Detection, Mitigation, and Criticality Analysis. *IEEE*.

J. Svacina, J. E. (2020). On Vulnerability and Security Log analysis: A Systematic Literature Review on Recent Trends. *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*.

Jain, N. G. (2023). Malicious URL: Analysis and Detection using Machine Learning. *International Conference on Computing for Sustainable Global Development*.

Jason Wiharja, A. K. (2024). *Peran Splunk Dalam Analisis Cyber Security Dan Monitoring Kejadian*. Diambil kembali dari inovation: https://innovation.co.id/peran-splunk-dalam-analisis-cyber-security-dan-monitoring-kejadian/

Jason Zurawski, J. M. (2023). National Institute of Standards and Technology Requirements Analysis Report. *OSTI.GOV*.

Kamil Kaczyński, M. G. (2020). Access Logs – Underestimated Privacy Risks. *International Journal of Electronics and Telecommunications*.

Kawahara, Y. (2020). Understanding the underlying algorithms and theories of machine learning. *Scimago Jurnal* .

Lab, I. (t.thn.). *PHP Webshell with handy features*. Diambil kembali dari b374k.php: https://github.com/b374k/b374k

Lau, R. (2021). Web Server Part 1: Apache/Nginx Basics. *semantic scholar* .

M. Selvaganesh, P. N. (2022). Efficient Brute-force handling methodology using Indexed-Cluster Architecture of Splunk. *IEEE*.

Magelang, P. (2024). *A01:2021-Broken Access Control*. Diambil kembali dari pustaka magelang: https://pustaka.magelangkota.go.id/books/keamanan-aplikasi-berbasis-web/page/a012021-broken-access-control

Mehta, D. (2021). Splunk Search Processing Language. *IEEE*.

Nashikkar, R. N. (2023). Enhancing Malicious Domain Detection Using Advanced Machine Learning Techniques. *IEEE*.

Networks, P. (2025). *What Is NIST?* Diambil kembali dari Paloalto Networks: https://www.paloaltonetworks.com/cyberpedia/nist

OWASP. (2021). *A01:2021 – Broken Access Control*. Diambil kembali dari OWASP Top 10:2021: https://owasp.org/Top10/A01_2021-Broken_Access_Control/

Prajkta khaire, R. N. (2022). Web Server Analysis. *International Journal For Science Technology And Engineering*.

Raditya Faisal Waliulua, S. T. (2020). Desain dan Implementasi Deteksi WebShell Malicious Web . *Jurnal Sistem Informasi Bisnis*.

Reddy, T. L. (2021). FORENSICS ON A HACKED WEBSITE. *INTERNATIONAL JOURNAL OF INNOVATIONS IN ENGINEERING RESEARCH AND TECHNOLOGY*.

Sadeeq Jan1, *. S. (2025). Access Restricted: A Study of Broken. *Archives of Advanced Engineering Science*.

Salfati, E. (2022). Digital Forensics and Incident Response(DFIR) Framework for Operational Technology (OT) . *NISTIR 8428*.

Salvationdata. (2024). *10 Techniques for Log File Analysis in Digital Forensics*. Diambil kembali dari salvation data technology: https://www.salvationdata.com/knowledge/log-file-analysis/

Security, I. (2020). *IBM X-Force Threat Intelligence Index 2024*. Diambil kembali dari Cyber Security Intelligence Index: https://www.ibm.com/security/data-breach/threat-intelligence

SentinelOne. (2025, Januari). *What is an Access Log? and How to analyze Access Logs?* Diambil kembali dari SentinelOne: https://www.sentinelone.com/cybersecurity-101/cybersecurity/what-is-an-access-log/

Shelmire, A. (2025). *Detecting Web Shells in HTTP Access Logs*. Diambil kembali dari anomali inc: https://www.anomali.com/blog/detecting-web-shells-in-http-access-logs

Shilin He, P. H. (2020). A Survey on Automated Log Analysis for Reliability Engineering. *ACM Computing Surveys (CSUR)*.

Sim Sang Gyoo, L. S. (2019). Method and apparatus for machine learning.

Splunk. (2023). *Search Processing Language (SPL) Overview*. Diambil kembali dari Splunk Documentation: https://docs.splunk.com/Documentation/Splunk

Splunk. (2025, May Monday). *Splunk Cloud Platform Admin Manual*. Diambil kembali dari Use the Search dashboards: https://docs.splunk.com/Documentation/SplunkCloud/9.3.2411/Admin/MonitoringSearch#Analyze_search_usage_statistics

Splunk. (2025). *Understanding SPL syntax*. Diambil kembali dari Splunk Cloud Platform: https://help.splunk.com/en/splunk-cloud-platform/search/search-reference/9.3.2411/introduction/understanding-spl-syntax

Splunk, E. (2025). *search modes*. Diambil kembali dari using the search app: https://help.splunk.com/en/splunk-enterprise/search/search-manual/9.2/using-the-search-app/search-modes

Srinivas. (2021, Januari). *Log analysis*. Diambil kembali dari Infosec: https://www.infosecinstitute.com/resources/digital-forensics/log-analysis/

Suheni, A. A. (2023). Penerapan Splunk Terhadap Analisis File Log Anomaly Keamanan. *Journal of Information Technology and society (JITS)* .

Xiaobo Yu, W. M. (2023). TridentShell: An enhanced covert and scalable backdoor injection attack on web application. *Journal of Network and Computer Applications*.

Yogi, I. R. (2019). Analisa log web server untuk mengetahui pola perilaku pengunjung website menggunakan teknik regular expressions. *Coding : Jurnal Komputer dan Aplikasi*.

Yusra Al Balushi(B), H. S. (2023). The Use of Machine Learning in Digital. *Modern College of Business and Science*.

Zhang Y, Z. J. (2020). Research on the detection of malicious domains based on machine learning methods. *IEEE International Conference on Information Technology, Big Data, and Artificial Intelligence*.