BAB II
LANDASAN TEORI

2.1 Penelitian Terkait

Penelitian yang dilakukan oleh (Wintolo et al., 2025) membahas mengenai analisis deteksi penyusup pada layanan *Open Journal System* (OJS) dengan menerapkan metode *DFRWS Investigative Model*. Penelitian ini dilatarbelakangi oleh tingginya risiko serangan siber pada platform publikasi ilmiah yang dapat mengancam integritas data penelitian. Hasil penelitian menunjukkan bahwa penggunaan metode DFRWS Investigative Model mampu mengidentifikasi jejak digital penyusup secara sistematis melalui tahapan identifikasi hingga pelaporan bukti.

Penelitian lain oleh (Khatib Sulaiman & Pakuan, n.d.) berfokus pada analisis keamanan website menggunakan standar *Open Web Application Security Project* (OWASP) Top 10. Penelitian ini menggarisbawahi bahwa banyak website institusi yang masih memiliki celah keamanan kritis pada sisi konfigurasi server dan manajemen akses. Dengan menggunakan metode OWASP, penelitian tersebut berhasil memetakan kerentanan utama yang sering dieksploitasi oleh penyerang untuk melakukan pencurian data.

Dalam penelitian yang dilakukan oleh (Wibisono, 2025), dibahas mengenai efektivitas analisis log server untuk mendeteksi serangan pada aplikasi web di lingkungan organisasi. Penelitian ini menekankan bahwa log server merupakan sumber bukti digital yang sangat valid karena mencatat setiap request secara detil,

sehingga mampu mendeteksi aktivitas mencurigakan yang sering kali tidak terdeteksi oleh sistem keamanan standar.

Selanjutnya, penelitian oleh (Lilhadiksi Razsanjani et al., 2024) melakukan analisis perbandingan performa antara web server berbasis Python dan Go pada protokol *Hypertext Transfer Protocol* (HTTP). Hasil pengujian menunjukkan bahwa bahasa pemrograman Go memiliki keunggulan signifikan dalam hal efisiensi penggunaan memori dan kecepatan pemrosesan data, terutama saat menangani beban trafik yang tinggi (*high concurrency*). Hal ini membuktikan bahwa Go sangat relevan digunakan untuk membangun alat bantu analisis data berskala besar seperti log parser.

Penelitian terkait lainnya oleh (Mubarok & Romli, 2024) serta (Gilvy Langgawan Putra et al., n.d.) sama-sama memanfaatkan perangkat lunak Wireshark untuk melakukan analisis trafik jaringan. Mubarok dan Romli memfokuskan pada deteksi serangan *Brute Force*, sementara Putra dkk. menggunakannya untuk metode sniffing dalam memvalidasi paket data. Kedua penelitian ini menunjukkan bahwa analisis paket data secara langsung memberikan tingkat akurasi yang lebih tinggi dalam memverifikasi sebuah serangan siber dibandingkan hanya mengandalkan laporan sistem.

Kajian terhadap penelitian-penelitian terdahulu lainnya seperti yang dilakukan oleh (Syaifudin et al., 2025), (Khatib Sulaiman & Pakuan, n.d.), serta (Pahlawansah et al., 2025) secara umum masih berfokus pada penilaian kerentanan (vulnerability assessment) dan pengujian keamanan website menggunakan standar OWASP.

Selain itu, penelitian dari (Wintolo et al., 2025), (Ramadhan et al., n.d.), serta (Afifah Rodhiyatun Nisa et al., 2024) memberikan penguatan pada aspek metodologi investigasi forensik jaringan dan pemanfaatan analisis log server untuk mendeteksi ancaman keamanan siber.

Berdasarkan kajian terhadap berbagai penelitian terdahulu tersebut, dapat diketahui bahwa analisis keamanan umumnya masih dilakukan secara terpisah antara analisis log aplikasi dan analisis paket jaringan. Selain itu, penggunaan bahasa pemrograman dengan performa tinggi seperti Go untuk membangun aplikasi log parser yang spesifik menangani layanan berbasis *Service Oriented Architecture* (SOA) di institusi pendidikan masih jarang dilakukan.

Penelitian ini memiliki perbedaan signifikan dibandingkan penelitian sebelumnya, yaitu pada pengembangan aplikasi log parser berbasis bahasa pemrograman Go yang diintegrasikan dengan metode DFRWS Investigative Model untuk menganalisis *malicious request* pada website SIAKAD Universitas Muhammadiyah Bengkulu yang mengadopsi arsitektur SOA. Penggunaan Go diharapkan mampu memberikan kecepatan proses analisis log yang lebih optimal, sementara verifikasi fungsionalitas melalui Wireshark akan memastikan mekanisme pattern matching berjalan sesuai spesifikasi.

2.2 Tinjauan Teoritis

2.2.1 Services Oriented Arhictecture (SOA)

Service Oriented Architecture (SOA) merupakan sebuah arsitektur teknologi informasi yang mendefinisikan cara untuk membuat unit-unit

layanan perangkat lunak agar dapat digunakan kembali (reusable) dan berinteraksi satu sama lain melalui standar protokol tertentu. Dalam lingkungan institusi pendidikan seperti Universitas Muhammadiyah Bengkulu, arsitektur SOA diterapkan pada Sistem Informasi Akademik (SIAKAD) untuk mengintegrasikan berbagai layanan data, mulai dari data kemahasiswaan, keuangan, hingga nilai akademik dalam satu ekosistem yang terintegrasi. Meskipun implementasi teknis SIAKAD UMB tidak menggunakan protokol web service formal seperti WSDL atau *REST API*, sistem ini mengadopsi prinsip-prinsip SOA secara konseptual melalui pemisahan modul layanan yang independen namun terintegrasi.

Pemanfaatan SOA memungkinkan setiap layanan (seperti layanan biodata mahasiswa atau layanan pengisian KRS) berfungsi secara independen namun tetap dapat bertukar informasi melalui mekanisme web service. Hal ini sejalan dengan teori manajemen sistem informasi yang menyatakan bahwa arsitektur yang fleksibel sangat krusial dalam mendukung operasional organisasi di era digital (Laudon & Laudon, 2018). Namun, implementasi SOA juga menghadirkan tantangan keamanan yang kompleks, karena setiap endpoint layanan yang terbuka menjadi target potensial bagi serangan siber (Gilvy Langgawan Putra et al., n.d.). Kerentanan pada satu layanan berbasis SOA dapat dieksploitasi untuk mengakses data sensitif pada layanan lainnya jika tidak dimonitoring dengan ketat (Ahmad et al., 2023)

2.2.2 Malicious Request

Malicious request adalah permintaan berbahaya yang dikirimkan oleh pihak tidak bertanggung jawab melalui protokol HTTP untuk mengeksploitasi kerentanan pada aplikasi web. Serangan ini sering kali menargetkan parameter URL atau input form dengan tujuan melakukan tindakan ilegal seperti pencurian data, pengambilalihan sistem, atau perusakan layanan. Deteksi dini terhadap pola permintaan ini sangat krusial guna mencegah kebocoran data sensitif institusi (Wibisono, 2025). Dalam konteks SIAKAD UMB yang mengadopsi prinsip SOA, setiap *endpoint* layanan yang terbuka berpotensi menjadi sasaran berbagai jenis malicious request. Secara umum, jenis-jenis serangan yang termasuk dalam kategori malicious request adalah sebagai berikut.

a. SQL Injection

SQL *Injection* adalah teknik serangan di mana penyerang menyisipkan atau memanipulasi perintah SQL melalui input pengguna yang tidak divalidasi, sehingga dapat mengakses, memodifikasi, atau menghapus data pada basis data secara ilegal. Penyerang umumnya memanfaatkan karakter khusus seperti kutip tunggal (%27), tanda komentar (--), serta keyword SQL seperti *UNION*, *SELECT*, dan *CAST* untuk mengubah logika query yang dieksekusi oleh aplikasi (Pahlawansah et al., 2025). Dalam penelitian ini, SQL *Injection* terdeteksi sebagai serangan dengan severity *HIGH* yang menargetkan langsung parameter

login SIAKAD UMB, mengindikasikan adanya upaya serius untuk menembus autentikasi sistem akademik.

b. Remote Code Execution (RCE)

Remote Code Execution (RCE) merupakan jenis serangan di mana penyerang berhasil mengeksekusi perintah atau kode program secara sewenang-wenang pada server target dari jarak jauh. Serangan ini umumnya dilakukan melalui eksploitasi *webshell* yang telah tertanam sebelumnya, atau melalui celah pada aplikasi web yang tidak memvalidasi input secara ketat (Viet et al., 2024). RCE dikategorikan sebagai serangan paling kritis karena jika berhasil, penyerang dapat mengambil alih kendali penuh atas server, mengunduh *malware*, mencuri data, atau menjadikan server sebagai bagian dari jaringan botnet. Dalam penelitian ini, pola RCE yang ditemukan menggunakan parameter *?cmd=* pada berbagai path *webshell* dengan payload eksekusi skrip eksternal dari domain gsocket.io. Pola RCE yang ditemukan dalam log SIAKAD UMB seluruhnya mendapat respons HTTP 404, mengindikasikan bahwa *endpoint webshell* yang dituju tidak ditemukan di server, sehingga eksploitasi tidak berhasil dilakukan.

c. Scanning / Reconnaissance

Scanning atau *reconnaissance* adalah fase awal dalam siklus serangan siber di mana penyerang melakukan pemetaan terhadap infrastruktur target sebelum melancarkan serangan yang lebih spesifik. Aktivitas ini mencakup percobaan akses ke berbagai path sensitif seperti panel administrasi, file

konfigurasi, dan direktori tersembunyi untuk menemukan celah yang dapat dieksploitasi(Khatib Sulaiman & Pakuan, n.d.). Meskipun aktivitas scanning sendiri belum secara langsung merusak sistem, informasi yang dikumpulkan pada fase ini menjadi dasar bagi penyerang untuk merencanakan serangan lanjutan yang lebih terarah. Kehadiran pola scanning dalam log backend mengindikasikan adanya *request* yang mencocoki signature pengintaian, namun karena keterbatasan log, tidak dapat dipastikan apakah berasal dari trafik eksternal atau internal infrastruktur.

d. Bot / Automated Scanner

Bot atau *automated scanner* adalah perangkat lunak otomatis yang dirancang untuk mengirimkan sejumlah besar request HTTP secara cepat dan sistematis guna mencari kerentanan pada aplikasi web target. Berbeda dari serangan manual, bot menggunakan *User-Agent* library HTTP seperti *python-requests* atau *Go-http-client* yang secara implisit mengidentifikasi dirinya sebagai skrip otomatis, bukan browser manusia(Lilhadiksi Razsanjani et al., 2024). Penggunaan bot memungkinkan penyerang melakukan enumerasi ratusan hingga ribuan endpoint dalam waktu singkat, termasuk mencari halaman registrasi, panel login, dan endpoint API yang tidak terlindungi. Kehadiran bot dalam jumlah signifikan pada log SIAKAD UMB menunjukkan adanya upaya pemindaian otomatis yang terorganisir.

e. Path Traversal

Path Traversal adalah serangan yang memanfaatkan karakter navigasi direktori seperti ../ untuk mengakses file atau direktori di luar *root* yang seharusnya diizinkan oleh aplikasi web. Tujuan serangan ini adalah membaca file sensitif sistem seperti */etc/passwd* pada sistem Linux atau file konfigurasi yang menyimpan kredensial basis data (Castagnaro et al., 2024). Jika aplikasi tidak melakukan sanitasi terhadap input path secara ketat, penyerang dapat menelusuri struktur direktori server dan mengekstrak informasi yang seharusnya bersifat privat.

f. Local File Inclusion (LFI)

Local File Inclusion (LFI) adalah serangan di mana penyerang memanipulasi parameter input yang digunakan aplikasi untuk memuat file lokal, sehingga file yang tidak seharusnya diakses dapat dieksekusi atau dibaca oleh server. LFI dapat dimanfaatkan untuk membaca file log, file konfigurasi, hingga mengeksekusi kode PHP berbahaya apabila penyerang berhasil menggabungkannya dengan teknik *log poisoning* (Ameen Noman et al., 2024). Dalam penelitian ini, salah satu temuan LFI bahkan menggunakan eksploitasi berbasis CVE yang telah dipublikasikan (CVE-2023-23333), mengindikasikan penggunaan tool eksploitasi terautomasi yang menargetkan kerentanan spesifik.

g. Cross-Site Scripting (XSS)

Cross-Site Scripting (XSS) adalah serangan injeksi di mana penyerang menyisipkan skrip berbahaya ke dalam halaman web yang kemudian dieksekusi oleh browser pengguna lain yang mengakses halaman tersebut.

Serangan XSS dapat digunakan untuk mencuri *session cookie*, melakukan *phishing*, atau mengarahkan pengguna ke situs berbahaya (Hannousse et al., 2022). Dalam konteks SIAKAD UMB, XSS yang berhasil berpotensi mengakibatkan pembajakan sesi login mahasiswa atau administrator, sehingga penyerang dapat mengakses data akademik tanpa perlu melakukan autentikasi.

2.2.3 OWASP Top 10

Open Web Application Security Project (OWASP) Top 10 adalah standar dokumentasi kesadaran global yang berisi daftar 10 risiko keamanan aplikasi web yang paling kritis. Standar ini digunakan dalam penelitian sebagai acuan untuk mengklasifikasikan jenis kerentanan yang ditemukan pada layanan berbasis SOA (OWASP Foundation, 2021). Fokus utama dalam penelitian ini adalah pada celah keamanan yang memungkinkan terjadinya manipulasi request dan kegagalan kontrol akses (Pahlawansah et al., 2025).

2.2.4 DFRWS Investigative Model

DFRWS Investigative Model adalah metodologi investigasi forensik jaringan yang bersifat progresif dan terstruktur untuk menangani insiden keamanan siber. Metode ini terdiri dari enam fase utama: *Identification, Preservation, Collection, Examination, Analysis, dan Presentation* (Wintolo et al., 2025). Keunggulan DFRWS Investigative Model terletak

pada kemampuannya dalam menjaga integritas bukti digital dari level jaringan hingga pelaporan yang valid (Khatib Sulaiman & Pakuan, n.d.).

2.2.5 Golang (GO)

Go atau Golang adalah bahasa pemrograman yang dikembangkan oleh Google dengan fokus pada efisiensi performa dan manajemen memori yang optimal. Fitur *concurrency* (*Goroutine*) pada Go memungkinkan aplikasi log parser memproses ribuan baris data log secara simultan dengan sangat cepat (Lilhadiksi Razsanjani et al., 2024). Hal ini menjadikan Go sebagai pilihan ideal untuk melakukan analisis *malicious request* dalam skala besar di SIAKAD UMB.

2.2.6 Wireshark

Wireshark adalah perangkat lunak penganalisis protokol jaringan (*packet sniffer*) yang mampu menangkap dan membedah trafik data secara real-time. Dalam penelitian ini, Wireshark digunakan sebagai instrumen verifikasi fungsionalitas untuk memastikan bahwa pola yang terdeteksi aplikasi Log Parser konsisten dengan paket data HTTP yang tertangkap pada simulasi lingkungan terkontrol (Mubarok & Romli, 2024).