BAB I

PENDAHULUAN

1.1 Latar Belakang

Aplikasi web menjadi komponen penting dalam berbagai sektor, termasuk dalam sektor pemerintahan, bisnis, dan juga penyedia layanan publik lainnya. Ketersediaan dan aksesbilitias yang bisa memudahkan dalam mencari informasi secara langsung dengan mudah dan efesien (Wahyu dkk., 2024). Seiring meningkatnya penggunaan aplikasi web, tentunya ancaman keamanan *cyber* juga semakin kompleks dan berkembang dengan banyak variannya. Pemindaian kerentanan (*Vulnerability Scanning*) dan *penetration testing (Pentest)* atau VAPT adalah cara agar mengetahui dan mendeteksi kerentanan yang terdapat pada aplikasi web (Almaarif & Lubis, 2020).

Berdasarkan daftar kerentanan OWASP Top 10 jenis kerentanan *SQL injection* masih tercatat sebagai ancaman yang paling serius bagi keamanan aplikasi web hingga saat ini (Saputra dkk., 2025). Menunjukkan bahwa jenis serangan ini masih menjadi ancaman yang paling umum sering ditemukan dan berbahaya untuk aplikasi web modern hingga saat ini (Alanda dkk., 2020). Jenis Kerentanan ini memungkinkan penyerang untuk menyisipkan perintah SQL berbahaya ke dalam kueri *database* melalui *input* yang tidak terfilter dengan baik, sehingga membuka peluang untuk mengakses *database* dalam sistem untuk mendapatkan kredensial penting yang ada di dalam *database* (Riyanti dkk,. 2024). *Blind SQL Injection* merupakah jenis *SQL injection* yang mencakup teknik *Boolean blind* dan *Time based blind*. Varian *Time Based* dikenal sulit dideteksi karena tidak menghasilkan

keluaran langsung, melainkan menunjukkan keberhasilan injeksi melalui penundaan waktu (*delay*). *Blind SQL Injection* memiliki tingkat kerumitan yang tinggi karena harus memasukan sintaks secara bertahap untuk menemukan celahnya dan Time Based Blind memasukan sintaks secara bertahap lalu menunggu respons delay untuk memvalidasi celahnya. (Nugroho & Mandala, 2020; Ombagi, 2019).

Time-Based Blind SQL Injection merupakan teknik yang relatif sangat lambat karena membutuhkan banyak permintaan (request) berulang ke server dan memiliki waktu eksekusi lebih tinggi dibandingkan *Boolean-Based* atau Teknik *SQL injection* lainya. Teknik ini bekerja dengan cara menebak data secara bertahap atau diuji satu persatu hingga menemukan karakter yang di inginkan lalu *delay* (Sommervoll dkk., 2024). Pendekatan teknik eksploitasi manual tentu tidak efektif untuk ekstploitasi *database* karena membutuhkan waktu yang lama, untuk meningkatkan efisiensi eksploitasi adalah dengan memanfaatkan algoritma pencarian, yang dapat mempercepat proses ekstraksi karakter *database* secara otomatis dan bertahap (Nugroho & Mandala, 2020). *HTTP header* dapat dijadikan media injeksi *time based SQLi* (Ombagi, 2019) mengembangkan alat berbasis *Python* yang menyisipkan *payload* pada *header* HTTP dengan memanfaatkan *delay* respons untuk mengekstraksi data dari *database* secara otomatis. Serta studi mengenai serangan berbasis waktu pada jaringan yang menekankan bahwa pengujian perlu diuji dalam berbagai kondisi waktu untuk mencerminkan perilaku server di lingkungan nyata(Sharon dkk., 2021).

Berbagai penelitian terdahulu telah membahas penerapan algoritma pencarian dalam proses eksploitasi *Blind SQL Injection* dan algoritma pencarian. (Nugroho &

Mandala, 2020) mengembangkan alat otomatisasi serangan *Blind SQL injection* dan menguji kinerja algoritma *Linear search, Binary search*, dan *Interpolation Search* untuk mengekstraksi *database* pada *Blind SQL Injection* yang berjenis *boolean based* di mana *Binary Search* terbukti memberikan hasil terbaik dengan waktu eksekusi (*runtime*) rata-rata 1,785 detik dibandingkan *Binary search* dan *linear search*, meskipun penelitian tersebut memiliki keterbatasan yaitu pengujian masih pada lingkungan *locallhost* membatasi representasi skenario dunia nyata, di mana *website* biasanya dihosting di server publik dengan berbagai konfigurasi jaringan dan keamanan.

Penelitian (Firnando Yusuf & Sijabat, 2023) berfokus pada jenis Serangan *Blind NoSQL Injection* melalui otomatisasi eksploitasi menggunakan algoritma *Linear search* dan *Binary search* pengujian dilakukan pada lingkungan *localhost* dengan target *database* MongoDB dan memanfaatkan rentang karakter ASCII sebagai ruang pencarian data. Hasilnya dari penelitian tersebut menunjukkan bahwa *Binary search* lebih unggul dan efesien dalam ekstraksi *database* pada skenario *blind SQL injection* yang berjenis *boolean based blind* pada parameter waktu eksekusi (*runtime*) dibandingkan *Linear search*.

Penelitian (Maurya & Singh, 2018)mengusulkan pendekatan *parallel sorting* yang menggabungkan *Ternary search* untuk meningkatkan efesiensi pencarian posisi elemen pada data. Pada penelitian tersebut, *Ternary search* digunakan untuk mempercepat proses penentuan indeks dengan membagi pencarian menjadi tiga bagian sehingga mengurangi jumlah langkah pencarian dibandingkan *Binary*

search. Hasilnya menunjukkan bahwa algoritma *Ternary search* mengurangi waktu eksekusi (*runtime*) pada ukuran skenario data 1k, 10k hingga 100k elemen.

Bedasarkan beberapa penelitian terdahulu, menunjukkan bahwa algoritma pencarian berperan penting dalam efisiensi eksploitasi *Blind SQL injection*, di mana *Binary Search* terbukti unggul dalam mengurangi *runtime* pada teknik *Boolean based blind* (Firnando Yusuf & Sijabat, 2023.; Nugroho & Mandala, 2020). Namun hasil tersebut belum dapat digeneralisasi pada *Time Based blind*, karena setiap *request* memicu *delay* sehingga performa bergantung pada percobaan query delay yang dikirimkan ke server. Penelitian ini membandingkan *Linear search* sebagai *baseline* yang umum pada inferensi karakter(Lutfina dkk., 2022). *Binary search* sebagai algoritma paling efisien pada studi terdahulu. Sedangkan *Ternary search* diterapkan untuk menguji apakah pembagian ruang pencarian menjadi tiga segmen (Maurya & Singh, 2018) dapat memberikan kinerja lebih baik ketika setiap percobaan memicu *delay*.

Penentuan parameter evaluasi sangat menentukan objektivitas penilaian efektivitas eksploitasi *Time Based blind*. Parameter *runtime* digunakan untuk mengukur kecepatan durasi waktu eksekusi proses ekstraksi dan telah menjadi metrik baku dalam beberapa penelitian terdahulu untuk mengukur kecepatan dan efesiensi eksploitasi(Firnando Yusuf & Sijabat, 2023.; Lutfina dkk., 2022; Nugroho & Mandala, 2020). Total *request* merupakan indikator yang krusial karena semakin banyak *request* yang di kirimkan berpotensi membebani server serta memperbesar potensi risiko deteksi oleh sistem keamanan mengaskan bahwa jumlah *request* mencerminkan efesiensi eksploitasi secara nyata. (Pružinec & Anh Quynh, 2023)

Berdasarkan penjelasan tersebut, terdapat gap penelitian karena kajian otomatisasi eksploitasi *Blind SQL Injection* sebelumnya yang masih berfokus pada teknik *Boolean-Based* dan belum mengevaluasi karakteristik *Time-Based Blind* yang jauh lebih kompleks dan sulit terdeteksi karena bergantung pada *delay* di setiap permintaan *payload,* sehingga kurangnya analisis yang menilai bagaimana perbedaan kompleksitas algoritma pencarian memengaruhi efesiensi proses eksploitasi (Firnando Yusuf & Sijabat, 2023.; Nugroho & Mandala, 2020 ;Maurya & Singh, 2018). Parameter evaluasi dalam penelitian terdahulu masih terbatas pada *runtime* tanpa mempertimbangkan total *request*, sedangkan *request* merupakan indikator krusial karena berkaitan dengan beban server dan mekanisme keamanan aplikasi (Pružinec & Anh Quynh 2023). Studi terkaait serangan berbasis waktu menunjukkan bahwa pengujian perlu dilakukan dengan variasi *delay* untuk mencerminkan kondisi server yang sebenarnya dan memininalkan bias waktu (Sharon dkk., 2021). Sementara penelitian terdahulu umumnya berbasis web *locall* atau *dummy* yang tinggi akan anomaly respons yang berbeda beda sehingga kurangnya representasi kondisi lingkungan nyata. (Nugroho & Mandala, 2020).

Penelitian ini melakukan evaluasi komparatif terhadap *Linear search*, *Binary search* dan *Ternary search* pada otomatisasi eksploitasi *Time Based blind sql Injection* untuk memperoleh gambaran performa algoritma yang lebih mencerminkan karakteristik kondisi aplikasi web pada lingkungan sebenarnya. Pengujian lakukan menggunakan dua parameter utama yaitu *runtime* dan total *request*, sehingga efektivitas setiap algoritma dapat dinilai lebih objektif pada kondisi dimana permintaan memicu delay. Melalui pendekatan ini, performa ketiga

algoritma dalam menangani berbagai skenario waktu *delay* pada *Time Based Blind* dapat dianalisis secara lebih terukur dan representatif terhadap kondisi aplikasi web nyata.

1.2 Rumusan Masalah

Berdasarkan penjelasan pada latar belakang, masalah dalam penelitian ini dapat dirumuskan sebagai berikut:

1. Bagaimana merancang otomatisasi eksploitasi *Time-Based Blind SQL Injection* pada aplikasi web dengan mengimplementasikan algoritma *Linear Search*, *Binary Search*, dan *Ternary Search*?
2. Bagaimana perbandingan performa ketiga algoritma pencarian tersebut dalam mengekstraksi *database* melalui pengujian *Time-Based Blind SQL Injection,* berdasarkan parameter *runtime* dan *request* pada berbagai skenario *delay?*

1.3 Tujuan Penelitian

Berdasarkan latar belakang dan rumusan masalah yang telah di paparkan, tujuan penelitian ini adalah:

1. Merancang otomatisasi eksploitasi *Time-Based Blind SQL Injection* pada aplikasi web dengan mengimplementasikan algoritma pencarian *Linear Search*, *Binary Search,* dan *Ternary Search*.
2. Menganalisis dan membandingkan performa algoritma *Linear search*, *Binary search*, dan *Ternary Search* dalam ekstraksi *database* melalui pengujian *Time-Based Blind SQL Injection*, berdasarkan parameter *runtime* dan total *requests* pada berbagai skenario *delay*.

1.4 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan manfaat secara akademis dalam bidang keamanan siber, khususnya pada kajian *Time-Based Blind SQL Injection* yang masih terbatas dalam literatur akademik dan memperkaya literatur mengenai penerapan algoritma pencarian dalam celah keamanan siber.

Secara praktis, penelitian ini memberikan manfaat bagi *pentester* dan pengembang aplikasi web dalam meningkatkan efisiensi pengujian keamanan serta masukan bagi pengelola untuk merancang strategi mitigasi yang sistematis.

1.5 Batasan Masalah

Penelitian ini dapat dilakukan secara terarah dan fokus, berikut adalah batasan-batasan masalah yang diterapkan:

1. Penelitian ini difokuskan pada jenis kerentanan *Time Based Blind SQL Injection* yang ditemukan pada aplikasi web.
2. Algoritma pencarian yang di jadikan pembanding dalam penelitian ini adalah *Linear search, Binary search* dan *Ternary search*.
3. Pengujian dilakukan pada aplikasi web yang telah memperoleh izin resmi untuk uji penetrasi.
4. Analisis performa pada dua parameter utama, yaitu *runtime* dan *total requests*, dengan variasi skenario *delay*.
5. Pengujian dilakukan pada *database* yang berjenis MySQL.
6. Pengujian dilakukan menggunakan *Blackbox Penetration Testing* tanpa akses ke kode sumber, hanya mengandalkan informasi eksternal seperti URL. Sehingga mitigasi hanya berbentuk rekomendasi saja.