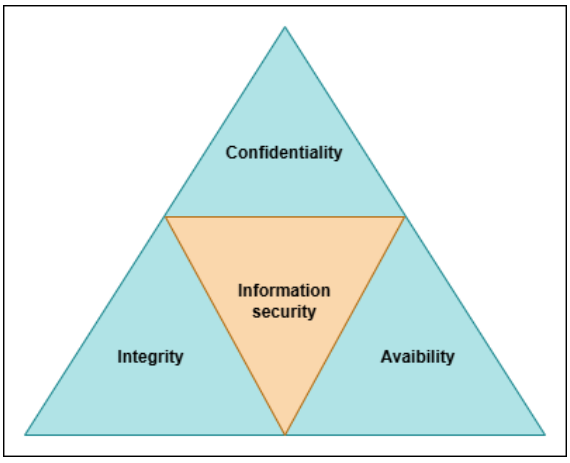
BAB II

LANDASAN TEORI

2.1 Keamanan Aplikasi Web

Keamanan Aplikasi web merupakan upaya melindungi keamanan sistem informasi dari akses pengguna yang tidak sah, modifikasi, serta gangguan yang dapat berdampak pada suatu organisasi atau instansi (De Paoli & Johnstone, 2023). Keamanan sistem informasi adalah upaya untuk menyelindungi informasi sebagai aset penting dari ancaman seperti penipuan, kebocoran, ataupun kegagalan sistem. Tujuannya adalah untuk mencegah atau mendeteksi upaya penipuan atau pertasan dalam sistem informasi serta menjaga kerahasiaan, integritas, dan juga ketersediaan informasi. Kegagalan dalam menjaga keamanan sistem informasi dapat menyebabkan kerugian finansial, penurunan produktifitas, dan dampak negative yang bisa mempengaruhi instansi atau perusahaan terkait (Nurul dkk., 2022). Keamanan informasi melibatkan tiga aspek utama dalam model *CIA Triangle Confidentiality*, *Integrity,* dan *Availability,* yang harus dilindungi dari berbagai ancaman dan risiko.



Gambar 2.1 CIA Triad (Harahap dkk., 2022.)

Gambar 2.1 merupakan gambar dari *CIA Triangle* yang mana merupakan aspek utama yang harus ada pada keamanan informasi yang meliputi sebagai berikut:

1. Kerahasiaan (*confidentiality*): menjamin bahwa informasi hanya dapat diakses oleh pihak yang bermenang dan sudah mendapat izin, lalu melindungi kerahasiaan data selama proses pengiriman, penerimaan, dan juga penyimpanan.
2. Integritas (*integrity*): memastikan keakuratan dan kelengkapan informasi dengan menerapkan sistem serta metode pengolahan data yang andal dan juga benar supaya menjagadla integritas dari sistem informasi itu tersebut.
3. Ketersediaan (*Availability*): memastikan informasi yang valid dan relevan dapat diakses oleh pihak yang membutuhkan ataupun masyarakat luas dalam lingkup website publik atau pemerintahan.

2.2 *Penetration Testing*

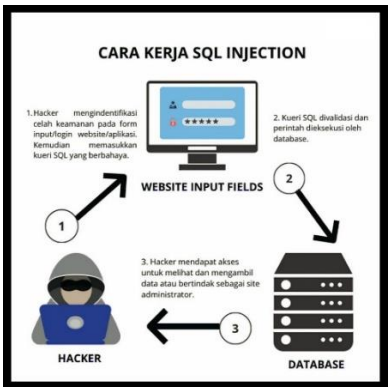
Pengujian Penetrasi *(Penetration Testing)* merupakan sebuah pengujian keamanan untuk mengekploitasi keamanan sistem informasi dengan cara mensimulasikan serangan dari pihak internal maupun eksternal terhadap aplikasi, *website*, jaringan, ataupun infrastruktur (Alam, 2023). Tujuan utama dilakukan *penetration testing* adalah untuk mengidentifikasi kerentanan keamanan, menguji kelemahan yang ada, dan memberikan rekomendasi perbaikan untuk memperkuat sistem (Faizi dkk., 2023).

Penetration terdiri dari 3 metode *Black Box, White Box,* dan *Grey Box*. *Blackbox* dilakukan tanpa informasi sistem apapun, seperti layaknya peretas di dunia nyata. *Whitebox* memberi akses penuh ke kode dan arsitektur untuk analisis

mendalam. *Greybox* adalah kombinasi keduanya dengan akses terbatas ke informasi internal. diaksanakan oleh pihak yang dengan perizinan dan tujuan untuk meningkatkan keamanan sistem dan mengetahui dampak dari kerentanan tersebut bukan untuk merusaknya (Kozel dkk., 2024).

2.3 *SQL Injection*

SQL Injection (SQLi) merupakan salah satu jenis kerentanan keamanan yang sering ditemukan di dalam aplikasi web, dimana serangan ini bekerja dengan cara menyisipkan kode berbahaya ke dalam perintah SQL melalui input yang tidak terfilter dengan baik. Teknik ini memungkinkan untuk memanipulasi *query database* sehingga dapat mengakses melihat atau bahkan menghapus data sensitif seperti kredensial pengguna, informasi pribadi, atau data rahasia lainnya (Yaswanthraj dkk., 2024). Meskipun *SQL Injection* memiliki fleksibilitas tinggi dalam pengelolaan basis data, kelemahan dalam pengamanan aplikasi yang tidak menerapkan validasi input dengan baik dapat dimanfaatkan oleh peretas untuk menjalankan perintah SQL secara tidak sah (Mukherjee & Sen., 2020). Gambar 2.2 merupakan Cara kerja *SQL injection* secara umum bekerja untuk mendapatkan akses ke *database* dan mengambil data pribadi yang ada pada aplikasi web target.



Gambar 2.2 Skema *SQL Injection* (Sari dkk., 2022.)

informasi yang terdapat pada ID-SIRTII/CC *(Indonesia Security Incident Response Team on Internet and Infrastructure/Coordination Center)* melalui situs resminya dimana perusahaan ini yang bekeja untuk menangani peringatan ancaman kemanan dan kerentanan suatu sistem atau jaringan seluruh Indonesia yang bekerja sama dengan MENKOMINFO (Kementerian Komunikasi dan Informatika). Berdasarkan data pada Tabel 2.1, jenis serangan yang paling sering terjadi adalah eksploitasi dan SQL, yang tercatat sebanyak 134.523 insiden dan SQL sebanyak 48839 insiden dari total keseluruhan serangan sejumlah 220.126 kasus.

Tabel 2.1 Data jenis serangan (Herman dkk., 2023)

Series	**29**	**30**	**31**	**01 Juni**	**02**	**03**	**04**	**Total Serangan**
EXPLOIT	7784	6151	16327	68176	12086	17471	6528	**134523**
SQL	6519	9988	9837	5814	5114	5537	6030	**48839**
DOS	3963	4034	3424	4351	4286	4186	3611	**27855**
DDOS	498	408	440	535	421	293	564	**3159**
BOTNET-CNC	82	173	91	74	162	1323	292	**2197**
BAD-TRAFFIC	153	147	179	141	138	163	182	**1103**
ORACLE	144	141	101	108	113	116	130	**853**
POLICY	45	131	39	74	85	196	140	**710**
SPYWARE-PUT	107	80	39	30	41	41	50	**388**
WEB-CLIENT	41	63	26	21	46	39	64	**300**

Series	**29**	**30**	**31**	**01 Juni**	**02**	**03**	**04**	**Total Serangan**
Other	27	24	24	40	30	30	24	**199**
Total Serangan	19363	21340	30527	79364	22522	29395	17615	**220126**

Berbagai jenis *SQL Injection* dapat diklasifikasikan ke dalam beberapa jenis berdasarkan metode atau cara eksploitasi dan respons yang di berikan oleh aplikasi web target. Berikut adalah beberapa tipe *SQL injection* yang umum di temukan pada aplikasi web berikut dengan kelebihan dan kekurangannya.

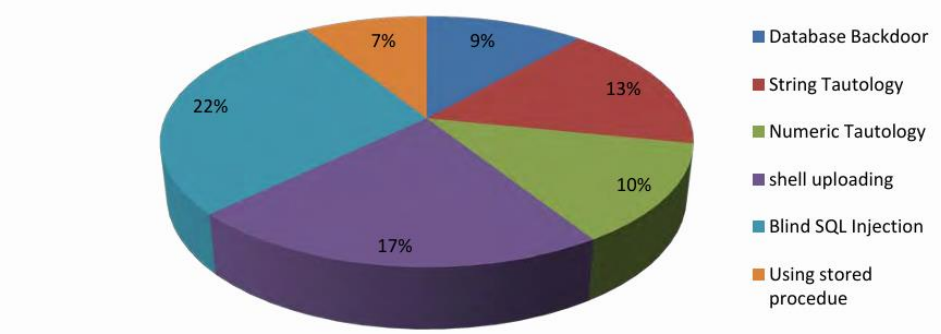
1. *Classic SQL Injection*: Serangan yang memanfaatkan *Query* SQL yang memberikan keluaran langsung dari *database* kepada penyerang web tersebut. Hasil yang di munculkan berupa pesan kesalahan yang di tampilkan langsung pada halaman web sehingga mempermudah untuk mengekstrak data namum pesan kesalahan yang muncul bisa dibatasi oleh pengembang (Azman dkk., 2021).
2. *Blind SQL Injection*: *Blind SQL injection* adalah teknik ini tidak memberikan keluaran atau umpan balik kueri yang di kembalikan kepada penyerang atau eksplisit dari *database* sehingga lebih sulit dideteksi dan sulit dieksploitasi secara manual tanpa bantuan alat otomatis *Blind SQL Injection* terbagi menjadi dua teknik yaitu *Time based Blind* dan *Boolean based blind* (Rosemond Dora dkk., 2023).
3. *Error- Based SQL Injection*: Teknik ini bekerja dengan mengeksploitasi pesan kesalahan yang diberikan oleh *database* untuk mendapatkan informasi sensitif pesan kesalahan dari *database* ini sering kali memberikan struktur *database*

atau informasi penting lainnya, teknik ini mudah di mitigasi dengan mengkonfigurasi *database* agar tidak menampilkan informasi sensitif. (Mondal dkk., 2022).

4. *Union-Based SQL Injection*: Teknik ini memanfaatkan perintah *union select* untuk menggabungkan hasil *query* yang valid dengan *query* yang tidak valid atau berbahaya yang dibuat oleh penyerang jika berhasil, penyerang dapat mengakses data yang tidak seharusnya terlihat (Yaswanthraj dkk., 2024).

2.4 *Blind SQL Injection*

Blind SQL Injection disebut serangan SQL "buta" karena terjadi ketika aplikasi tidak memberikan keluaran langsung atau pesan kesalahan dari *database* kepada penyerang. Dalam serangan ini penyerang biasanya mencoba mendapatkan informasi sensitif atau memodifikasi data dengan mengeksploitasi rentanan pada aplikasi web (Pružinec & Anh Quynh, 2023). *Query* SQL dikirimkan kepada *database*, penyerang dapat menentukan apakah suatu kondisi benar atau salah. Serangan ini cenderung fungsi deteksi dan ditangkal karena tidak ada umpan balik dari *database,* ini mengapa jenis ini biasanya dilakukan eksploitasi menggunakan alat otomatis. *Blind SQL Injection* terbagi menjadi dua metode utama yaitu *Time based Blind SQL Injection* dan *Boolean Based Blind SQL Injection*(Rosemond Dora dkk., 2023) .



Gambar 2.3 Presentase serangan Aplikasi web (Kaur, 2021.)

Merujuk pada Gambar 2.3 menunjukkan berbagai teknik eksploitasi keamanan dimana *Blind SQL injection* memiliki presentasi paling tinggi (22%), yang menandakan bahwa serangan *Blind SQL Injection* masih menjadi serangan yang paling umum yang ada pada aplikasi web. dibandingkan dengan metode lain, serangan ini lebih sulit dideteksi tetapi tetap efektif dan mengakses data sensitif.

2.4.1 *Time Based blind SQL Injection*

Metode ini merupakan metode yang memanfaatkan waktu tunda (*delay*) sebagai indikator keberhasilan eksekusi *query* SQL. Teknik ini digunakan ketika aplikasi web tidak bisa memberikan pesan kesalahan atau keluaran jelas yang dapat dimanfaatkan oleh penyerang untuk mengeksploitasi *database* (Johny dkk., 2021). Metode ini bekerja dengan penyerang mengirimkan *query* yang menyebabkan server menunda respons dalam jangka waktu tertentu jika kondisi dalam *query* terbukti benar. Sebagai contoh, penyerang dapat menyisipkan perintah IF (*condition*, SLEEP(n detik), 0) dalam *query* SQL. Jika kondisi yang diuji benar, *database* akan menunda respons selama n detik; namun, jika kondisi salah, respons akan diberikan segera tanpa penundaan waktu. (Rai dkk., 2021). Meskipun metode ini efektif, *Time-Based Blind SQL Injection* memerlukan waktu yang lebih lama

dibandingkan metode *SQL Injection* lainnya karena penyerang harus menjalankan sejumlah besar *query* dengan pengukuran waktu yang presisi (Avireddy dkk,. 2021)

2.5 *Penetration Testing Execution Standard (PTES)*

Penetration Testing Execution Standard (PTES) Pertama kali dikembangkan pada tahun 2009 untuk menyelaraskan pemahaman diantara layanan keamanan dan bisnis atau suatu instansi dalam pengujian penetrasi, dipimpin oleh Chris Nickerson dan Dave Kennedy, Standar ini bertujuan memberikan pedoman yang jelas bagi perusahaan ataupun instansi dalam menentukan cakupan dan manfaat pengujian penetrasi dengan adanya PTES, bisnis atau instansi dapat memahami jenis pengujian yang dibutuhkan sementara profesional keamanan mendapatkan panduan untuk memberikan layanan yang lebih efektif dan juga berkualitas (I Made & Gede 2024.).

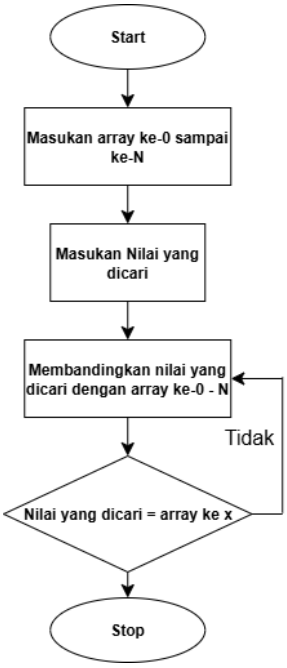
Penetration Testing Execution Standard (PTES) framework yang dirancang untuk memberikan panduan sistematis dalam melakukan pengujian penetrasi terhadap sistem informasi atau aplikasi web. PTES mencakup seluruh tahapan yang ada pada penetration testing, mulai dari perancangan hingga perlaporan hasil uji keamanan, sehingga memastikan pendekatan yang struktur dan metodologis dalam mengevaluasi keamanan sistem. (Tahir & Risky, 2024) Kerangka kerja ini mencakup tujuh tahapan utama, yaitu *Pre-engagement Interactions, Intelligence Gathering, Threat Modeling, Vulnerability Analysis, Exploitation, Post Exploitation,* dan *Reporting* (Safitra & Lubis, 2023).

2.6 Algoritma Pencarian

Proses pencarian (*searching*) merupakah salah satu elemen penting dalam pengolahan data. Tujuan utama dari algoritma pencarian dalah untuk menemukan suatu nilai atau data tertentu di dalam sekumpulan data dengan tipe yang seragam, baik berupa data sederhana maupun data yang kompleks. Secara umum, algoritma pencarian adalah algoritma yang menerima sebuah kunci sebagai masukan. Kemudian menjalankan serangkaian langkah untuk menemukan data yang sesuai dengan kunci tersebut. Hasil dari proses ini dapat berupa data yang ditemukan, ataupun tidak ditemukan (Ramadhan & Avrilia Lantana, 2023).

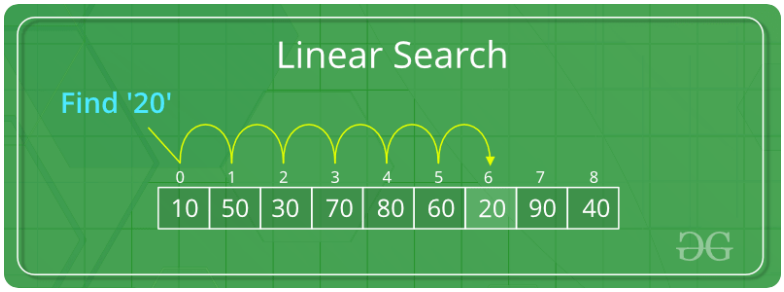
2.6.1 Algoritma *Linear Search*

Algoritma *Linear Search* adalah algoritma pencarian berurutan yang digunakan untuk menemukan data pada aray satu dimensi dengan memeriksa elemen dari awal sampai akhir atau dari 0 sampai N. Algoritma ini mampu bekerja pada kumpulan data yang terurut maupun tidak terurut keuntungan dari algoritma ini adalah jika data yang dicari berada di bagian awal aray maka data tersebut dapat segera ditemukan oleh algoritma tersebut(Abdelhamid dkk., 2023). Pada gambar 2.4 berikut adalah *flowchart* dari algoritma *Linear search*.



Gambar 2.4 Flowchart *Linear Search* (Lutfina dkk., 2022)

Gambar 2.4 merupakan proses pencarian dengan metode linear dimulai dengan menyiapkan array dari indeks ke-0 hingga ke-N sebagai ruang pencarian, Selanjutnya, nilai atau data yang ingin dicari dimasukkan sebagai input. Algoritma kemudian membandingkan nilai tersebut dengan setiap elemen array secara berurutan, dimulai dari indeks pertama (ke-0) hingga elemen di mana nilai yang sesuai berhasil ditemukan. Berikut adalah cara kerja *linear search* secara umum (Tere dkk., 2024).



Gambar 2.5 Cara kerja *Linear Search* (Tere dkk., 2024)

Pada Gambar 2.5 memperlihatkan contoh penerapan algoritma linear untuk mencari sebuah bilangan dalam array. Pada gambar 2.5 tersebut terdapat array dari indeks 0 hingga 8. Bilangan yang menjadi target pencarian adalah 20, sehingga proses dimulai dengan membandingkan angka 20 dengan elemen array dari indeks 0 secara berurutan hingga ditemukan kecocokan pada indeks ke-6 (Luo dkk., 2021). Untuk merepresentasikan proses pencarian secara formal, algoritma *linear search* dapat dinyatakan dalam bentuk formula matematis sebagai berikut.

$$(\boldsymbol{A}, \boldsymbol{x}) = \begin{cases} \boldsymbol{i}, & \boldsymbol{Jika\ A[i] = x} \\ \boldsymbol{-1} & \boldsymbol{jika\ x\ tidak\ di\ temukan} \end{cases}$$

Kompleksitas Linear Search dinyatakan sebagai berikut:

$$\boldsymbol{O(n)}$$

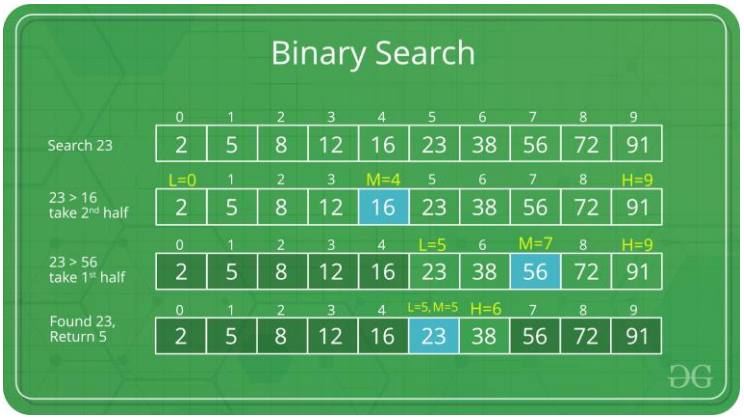
Formula pada Linear search menggambarkan bahwa pencarian dilakukan secara berurutan dari elemen pertama higga terakhir sampai ketemu karakter yang dicari.

2.6.2 Algoritma *Binary Search*

Algoritma Pencarian *Binary Search* merupakan algoritma pencarian pada aray yang elemen-elemennya telah dirutkan Proses pencarian dilakukan dengan membagi aray menjadi dua bagian secara berulang hingga data yang dicari ditemukan. Jika tidak, Daftar dibagi dua dan pencarian dilanjutkan pada bagian yang mungkin berisi data. ini diulang sampai data ketemu(Singh dkk., 2025). Gambar 2.6 berikut adalah *flowchart* dari *algoritma Binary search*. Berikut adalah cara kerja *Binary search* secara umum.

Gambar 2.6 *Flowchart Binary Search* (Firnando Yusuf & Sijabat, 2023.)

Gambar 2.6 *Binary search* bekerja dengan cara melihat elemen di tengah aray terlebih dahulu jika nilai yang dicari sama dengan elemen tengah maka pencarian selesai dan indeks elemen tersebut menjadi jawabannya jika nilai yang dicari lebih besar maka pencarian hanya diakuan di bagian kanan aray begitu pun sebaliknya jika nilai lebih kecil pencarian dilanjutkan di bagian kiri aray langkah ini terus diulang sampai nilai yang dicari ditemukan. Berikut adalah cara kerja *Binary search* secara umum (S. Kumar dkk., 2022).



Gambar 2.7 Cara kerja *Binary Search* (Mi dkk., 2022.)

Berdasarkan pada Gambar 2.7 terdapat aray dengan indeks 0 hingga 9 yang berisi bilangan acak Nilai yang akan dicari adalah 23. Langkah pertama yang diakukan dengan menentukan elemen di tengah dari aray tersebut. Diketahui bahwa nilai pada indeks tengah adalah 16. Karena 23 lebih besar daripada 16 Maka pencarian dilanjutkan pada bagian subray di sisi kanan Proses yang sama diakukan dengan mencari elemen tengah dari subray tersebut dan diulangi hingga ketemu nilai 23 (Mi dkk., 2019.) . Secara sistematis proses penentuan titik tengah (mid) dalam algoritma *binary search* dinyatakan sebagai berikut:

$$\boldsymbol{mid} = \lfloor \frac{\boldsymbol{Low} + \boldsymbol{high}}{\mathbf{2}} \rfloor$$

Dengan pembaruan batas pencarian sebagai berikut:

$$\textbf{Jika } \boldsymbol{x < A[mid]}, \boldsymbol{high = mid - 1}$$

$$\textbf{Jika } \boldsymbol{x > A[mid]}, \boldsymbol{low = mid + 1}$$

Kompleksitas Algoritma *Binary Search* adalah:

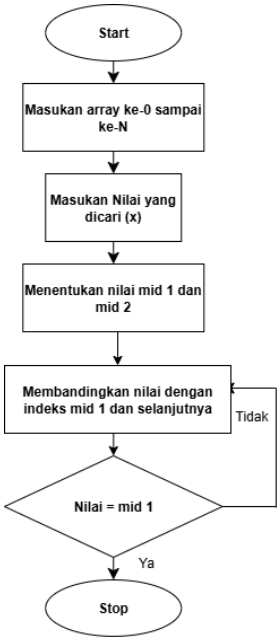
$$\boldsymbol{O}(\mathbf{log}\ \boldsymbol{n})$$

Formula atau Rumus ini menunjukkan *Binary Search* membagi ruang menjadi dua bagian pada rentang karakter yang akan di cari.

2.6.3 Algoritma *Ternary search*

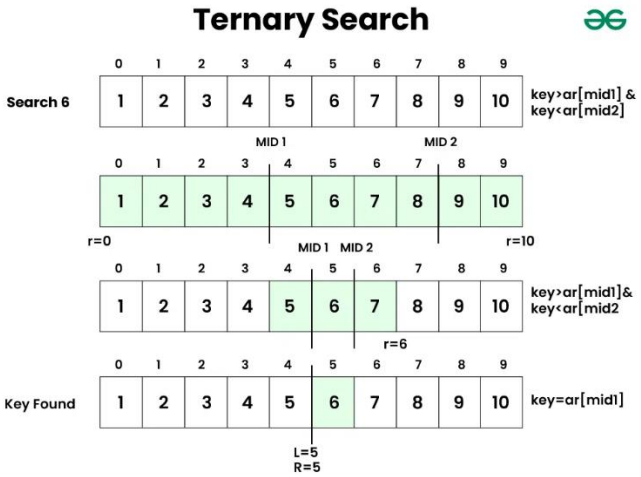
Algoritma Ternary search adalah algoritma berbasis divide dan conquer yang mana dirancang untuk mencari elemen tertentu dalam array. Cara kerjanya hampir sama dengan *Binary search* tetapi algoritma ini membagi menjadi 3 bagian bukan 2 bagian seperti binary search kemudian dilakukan pemeriksaan untuk menentukan

bagian mana yang mengandung kunci atau elemen yang dicari (Peng & Gao, 2024). Pada gambar 2.8 berikut adalah *flowchart* dari algoritma *Ternary search.*



Gambar 2.8 *Flowchart Ternary search* (Peng & Gao, 2024)

Gambar 2.8 merupakan proses algoritma *ternary search*, pencarian dimulai dengan membandingkan nilai target terhadap dua titik tengah, mid1 dan mid2. Jika nilai target sama dengan elemen di mid1, maka pencarian berakhir di sana. Jika belum, elemen di mid2 akan diperiksa. Apabila nilai yang dicari sesuai dengan mid2, hasil pencarian langsung ditemukan. Ketika nilai target lebih kecil dari mid1, pencarian dilakukan di bagian kiri array. Jika lebih besar dari mid2, proses diarahkan ke bagian kanan array. Namun, jika nilainya berada di antara kedua titik tengah tersebut, fokus pencarian hanya dilakukan pada bagian tengah (Oleh & Dwiyanto, 2020.).



Gambar 2.9 cara kerja *Binary search* (Oleh & Dwiyanto, 2021.)

Gambar 2.9 merupakan contoh dari cara kerja *ternary search* secara umum dimana pada Gambar 2.9 terdapat sebuah array dengan angka 1 sampai 10, dari array tersebut angka yang dicari adalah angka 6. Langkah awal adalah dengan cara membagi sebuah panjang array menjadi 3 bagian lalu tentukan mid1 dan mid2. setelah itu, nilai pada indeks diperiksa terhadap mid1 dan ternyata lebih besar. Lalu dibandingkan dengan mid2, dan kali ini hasilnya lebih kecil, dengan demikian, pencarian dilanjutkan hanya pada bagian di antara mid1 dan mid2, sampai akhirnya angka 6 ditemukan (Peng & Gao, 2024). Secara formula matematis, rumus penentuan kedua titik tengah dinyatakan sebagai berikut:

$$\boldsymbol{mid1 = Low + \lfloor \frac{high - low}{3} \rfloor}$$

$$\boldsymbol{mid2 = high - \lfloor \frac{high - low}{3} \rfloor}$$

Aturan pembaruan batas ruang pencarian sebagai berikut:

$$\textbf{Jika } \boldsymbol{x < A[mid1]},\ \boldsymbol{high = mid1 - 1}$$

$$\textbf{Jika } \boldsymbol{x > A[mid2]},\ \boldsymbol{low = mid2 + 1}$$

$$\textbf{Jika } \boldsymbol{A[mid1] < x < A[mid2]},\ \boldsymbol{low = mid1 + 1},\ \boldsymbol{high = mid2 - 1}$$

Kompleksitas Algoritma *Ternary Search* adalah:

$$\boldsymbol{O(\log_3 n)}$$

Formula atau Rumus ini menunjukkan *Ternary Search* membagi ruang menjadi tiga bagian pada rentang karakter yang akan di cari.

2.7 *Database*

Database atau basis data merupakan kumpulan informasi yang disimpan pada komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk mengakses atau memperoleh informasi dari *database* tersebut. *Software* yang digunakan untuk mengolah dan memanggil *query* yang ada pada *database* disebut sistem manajemen *database* atau *database* management system dbms. *Database* merupakan komponen utama yang paling penting yang ada pada sistem informasi karena berisikan informasi untuk pengambilan keputusan berasalkan dari *database* itu tersebut (Ahmadar dkk., 2021).

Jenis *database* MySQL merupakan *database* yang paling banyak digunakan pada sistem informasi atau aplikasi web karena jenis *database* MySQL ini terkenal dengan daya tahannya yang kuat dan cukup stabil untuk digunakan menyimpan data pribadi atau informasi yang ada pada aplikasi web (Ahmadar dkk., 2021). Selain jenis *database* MySQL, ada banyak jenis *database* lainnya yang sering digunakan seperti Oracle dan Postgre SQL (*Database new*, t.t.) yang biasanya digunakan pada aplikasi web atau sistem informasi lainnya.

2.8 *State of the art*

Penelitian ini diawali dengan studi literatur untuk mengkaji dan merangkum hasil penelitian sebelumnya yang memiliki keterkaitan dengan topik yang akan diteliti. Studi ini bertujuan untuk memperoleh pemahaman yang komprehensif mengenai perkembangan terbaru dalam bidang yang dikaji, serta mengidentifikasi kesenjangan penelitian yang belum terjawab. Penelitian ini dapat menyoroti *state of the art* yang menjadi dasar pengembangan penelitian lebih lanjut. Hasil studi literatur dirangkum dalam Tabel 2.2, yang menyajikan temuan utama dari penelitian terdahulu serta bagaimana penelitian ini memberikan kontribusi baru dalam bidang yang diteliti.

Tabel 2.2 State of the art

Peneliti	Judul Penelitian	Algoritma/ Metode	Hasil dan temuan	Kekurangan
(Pružinec & Anh Quynh, 2023)	*Hakuin: Optimizing BSQLI with Probabilistic Language Models*	*Binary Search* dan *Language model*	Hakuin 6x lebih efisien dari alat BSQLI lain; 3.2x lebih cepat dalam data generik; 26x pada kolom terbatas	Model bahasa Hakuin berbasis data Stack Exchange sehingga akurasi prediksi karakter bergantung pada kesesuaian konteks *database* target. Selain itu, penelitian ini tidak mengukur parameter kritikal untuk evaluasi kerentanan.
(Firnando Yusuf & Sijabat, 2024)	Optimasi Serangan Blind NoSQL Injection dengan Pendektakan Algoritma Binary Search	*Linear Search dan Binary Search*	Mengembangkan alat eksploitasi otomatis untuk Blind NoSQLi. Binary search terbukti 20–30% lebih cepat dari linear search pada skenario pengujian.	Penelitian hanya pada NoSQL Injection, bukan SQLi . Pengujian dilakukan secara lokal dan paramater keberhasilan pad runtime dan iterasi.
(Lutfina dkk., 2022)	*Comparative Analysis of Recursive and Iterative Methods in Linear Search Algorithms*	Linear Search 2 varian	Linear search terbukti jauh lebih efisien dari rekursif. iteratif 0.0133s vs rekursif 28.38s. Memori juga lebih hemat.	Fokus hanya pada konteks list searching konvensional, bukan eksploitasi SQLi. Tidak meneliti *delay*

Peneliti	Judul Penelitian	Algoritma/ Metode	Hasil dan temuan	Kekurangan
(Ramadhan & Avrilia Lantana, 2023)	Perbandingan Binary dan Sequential Search pada Web Inventory	*Binary search dan Linear Search*	Algoritma *Binary search* unggul di waktu eksekusi (rata-rata: 0.04ms vs 0.05ms; best-case: 0.0125ms vs 0.1129ms) dalam pencarian stok barang.	Hanya membahas efesiensi algoritma pencarian dan parammeter keberhasilan hanya pada waktu eksekusi saja
(Alanda et al., 2023)	*Web Application Penetration Testing Using SQL Injection Attack*	*Blackbox testing* dengan menggunakan *tools Metasploit dan hydra*	Hasil dan temuan dari penelitian ini menunjukkan bahwa 80% dari situs web yang diuji memiliki kelemahan terhadap serangan *SQL Injection.*	Kurangnya eksplorasi mendalam tentang teknik sepesifik untuk mecegah serangan *SQL Injection.* Penelitian ini hanya berfokus kepada identifikasi kerentanan
(Nugroho & Mandala, 2020)	*Study the Best PenTest Algorithm for Blind SQL Injection Attacks*	*Linear search,Binary search,Interpolation search*	Berhasil mengembangkan otomatisasi serangan Blind *SQL Injection* dan mengukur kinerja algoritma pencarian.	Lingkungan pengujian lokal dan jenis *blind SQL injection* tidak spesifik serta parameter pengujian di fokuskan pada *runtime*
(Erdodi dkk., 2021)	*Simulating SQL Injection Vulnerability Exploitation Using Q-Learning Reinforcement Learning Agen*	*Reinforcement Learning*	Agen dapat belajar mengeksploitasi SQLi secara otonom melalui interaksi iteratif. Eksperimen menunjukkan bahwa RL efektif dalam simulasi eksploitasi SQLi.	*Reinforcement Learning Agen* tidak di desain untuk ekstraksi data, Hanya diuji dalam lingkungan simulasi (bukan aplikasi nyata)

Peneliti	Judul Penelitian	Algoritma/ Metode	Hasil dan temuan	Kekurangan
(Aliero dkk., 2020)	*An algorithm for detecting SQL injection vulnerability using black-box testing*	*Black-box testing* + *SQLIVS* (scanner dengan komponen crawling, attack, analysis)	Dapat mendeteksi blind SQLi (tautology, union, order-by). Hasil uji pada 3 aplikasi menunjukkan recall dan F-measure tinggi dibanding *tools* lain.	Tidak akurat dalam mendeteksi kerentanan Time based Blind SQL i dan fokus pada deteksi bukan eksploitasi
(Ombagi, 2017)	*Time-Based Blind SQL Injection via HTTP Headers: Fuzzing and Exploitation.*	Teknik eksploitasi *time-based blind delay* via *HTTP header fuzzing*	Penggunaan header meningkatkan efektivitas *payload* enumeration dan fleksibilitas *payload delay*	Hanya memfokuskan pada teknik fuzzing header HTTP untuk memicu *delay* time-based blind SQLi. belum menilai bagaimana variasi *delay* (jumlah detik sleep) memengaruhi jumlah *request* atau efektivitas deteksi dan ekstraksi data
(Jibril Ibrahim -, 2024)	*Complexity Analysis between Bruteforcing and Blind SQL Injection with LIKE-Based Data Exfiltration*	*Brute forced vs Like based*	LIKE-based SQL Injection jauh lebih efisien dibanding brute force dalam meng ekstraksi data (kompleksitas linear dibanding eksponensial)	Penelitian ini lebih menekankan analisis kompleksitas algoritmik (*Big-O notation*) pada metode *brute force* dan *LIKE-based SQLi*, tanpa evaluasi komprehensif terhadap kinerja praktis, seperti jumlah permintaan (*requests*)
(Rorong dkk., 2025)	*Optimizing Search Efficiency in Ordered Data: A*	*Hybrid Jump Search + Binary Search (Jump*	JBS lebih efisien dibanding Jump Linear Search (JLS) pada data berurutan; eksekusi 0.008–	JBS hanya diuji pada dataset terurut dan statis tidak relevan untuk skenario ekstraksi databse dan Hasil hanya

Peneliti	Judul Penelitian	Algoritma/ Metode	Hasil dan temuan	Kekurangan
	Hybrid Approach Using Jump Binary Search	*Binary Search – JBS)*	0.012 ms (400 elemen) dan tetap stabil pada 0.010–0.020 ms (3200 elemen). Ternary Search menunjukkan performa terbaik pada dataset besar.	menyajikan waktu eksekusi untuk data lokal.
(Oleh & Dwiyanto, 2020.)	Analisis Perbandingan Algoritma *Ternary* dan *jump search*	*Ternary Search dan Jump Search*	Perbandingan dilakukan berdasarkan *runtime* dan *memory consumption,* dengan hasil bahwa *Ternary Search* cenderung lebih efisien pada dataset tertentu.	Tidak menguji performa algoritma di lingkungan terganggu latensi atau noise jaringan paramater hanya pada waktu eksekusi dan penggunaan memory
(Zhang dkk., 2019)	*The ART of SQL Injection Vulnerability Discovery*	*Adaptive Random Testing (ART)*	ARTSQLi mampu meningkatkan coverage pengujian dengan membangkitkan input acak yang lebih efektif untuk mendeteksi *SQL Injection*	Penelitian ART4SQLi lebih menekankan pada identifikasi potensi kerentanan SQL Injection melalui pendekatan *Adaptive Random Testing (ART)*, dan tidak mengukur parameter yang krusial
(Natanael, 2023)	Web Penetration dalam mencari kerentanan SQL injection	Penetration Testing dan OWASP Zap	Penelitian ini berhasil mengidentifikasi kerentanan SQL Injection pada website dummy khusus SQL i, OWASP JUICE SHOP. Berhasil	Metodologi yang digunakan mungkin tidak mencakup semua aspek pengujian keamanan yang diperlukan untuk analisis yang lebih komprehensif .

Peneliti	Judul Penelitian	Algoritma/ Metode	Hasil dan temuan	Kekurangan
			memperoleh hak akses admin dengan *payload* yang sederhana	
(Hajar Ismail dkk., 2024)	*A REVIEW OF PENETRATION TESTING PROCESS FOR SQL INJECTION ATTACK*	*BlackBox Peneteraation testing*	Memberikan kerangka metodologis dan tahapan sistematis dalam penetration testing *SQL Injection*, termasuk pembahasan tentang serangan *time-based*. Menunjukkan peran penting *tools* otomatis untuk eksploitasi umum SQLi.	Belum mengkaji secara teknis mekanisme eksploitasi *time-based blind SQL injection* secara mendalam, khususnya dalam konteks inferensi data berbasis penundaan waktu (*delay-based inference*). Terdapat keterbatasan dalam variasi alat otomasi pengujian
(Nugraha dkk., 2024)	SQL Injection: Analisis Efektivitas Uji Penetrasi dalam Aplikasi Web	Tiga Alat Uji penetrasi yang bersifat *Open Source*	Menemukan variasi kerentan SQL Injection pada 10 situs dengan situs B, I, dan J paling rentan akibat teknik pengembangan prosedural. Sementara itu situs C dan H lebih aman.	Keterbatasan dalam variasi alat pengujian dan sampel situs web yang terbatas. Hal ini menyarankan perlunya pendekatannya lebih inklusif dan diversifikasi alat pengujian untuk memperoleh perspektif yang lebih luas dan spesifik

Berdasarkan pada Tabel 2.2 State of the Art menunjukkan bahwa Berbagai penelitian terdahulu telah mengembangkan metode untuk mendeteksi dan mengeksploitasi *SQL Injection* maupun *Blind SQL Injection* menggunakan pendekatan seperti black-box testing, serta penerapan algoritma pencarian seperti *Linear Search* dan *Binary Search* yang terbukti mampu mempercepat proses ekstraksi data (Nugroho & Mandala, 2020; Pružinec & Anh Quynh, 2023). Sebagian besar penelitian tersebut masih terbatas pada konteks *Boolean-Based SQL Injection* dan dilakukan dalam lingkungan lokal yang tidak merepresentasikan kondisi jaringan nyata. Kajian terhadap *Time-Based Blind SQL Injection* (TBB-SQLi) yang lebih kompleks dan sulit dideteksi masih sangat minim, sementara aspek evaluasi efisiensi eksploitasi belum mempertimbangkan parameter penting seperti jumlah *request* dan variasi *sleep delay* yang berpengaruh langsung terhadap beban server dan efektivitas serangan. Kekurangan ini penting untuk diatasi karena tanpa analisis menyeluruh terhadap parameter tersebut, sulit menentukan algoritma pencarian yang paling efisien dalam skenario serangan nyata. Oleh Penelitian ini mengusulkan pengembangan sistem otomatisasi eksploitasi TBB-SQLi berbasis *Python* serta melakukan perbandingan performa tiga algoritma pencarian *Linear Search, Binary Search,* dan *Ternary Search* berdasarkan parameter *runtime* dan total *request* pada berbagai variasi *delay*, guna menghasilkan analisis komprehensif terhadap efisiensi dan efektivitas eksploitasi serta memberikan kontribusi teknis dalam mitigasi keamanan aplikasi web.

Tabel 2.3 Matriks Penelitian

Judul Penelitian	Ruang Lingkup Penelitian										
	Blind SQL i		Algoritma			Tujuan		Objek		Parameter Evaluasi	
	Time Blind SQLi	Boolean Blind SQLi	Linear Search	Binary Search	Ternary Search	Deteksi	Eksploitasi	Web online	Local	Runtime	*Request*
Hakuin: Optimizing BSQLI with Probabilistic Language Models (Pružinec & Anh Quynh, 2023)		✓		✓			✓				✓
Optimasi Serangan Blind NoSQL Injection dengan Pendektakan Algoritma Binary Search. (Firnando Yusuf & Sijabat, 2024.)			✓	✓			✓		✓	✓	
Comparative Analysis of Recursive and Iterative Methods in Linear Search Algorithms. ((Lutfina dkk., 2022)			✓			✓			✓	✓	
Perbandingan Binary dan Sequential Search pada Web Inventory. (Ramadhan & Avrilia Lantana, 2023)			✓	✓		✓		✓		✓	
Web Application Penetration Testing Using SQL Injection Attack. (Alanda et al., 2023)		✓				✓		✓			

Judul Penelitian	Ruang Lingkup Penelitian										
	Blind SQL i		Algoritma			Tujuan		Objek		Parameter Evaluasi	
	Time Blind SQLi	Boolean Blind SQLi	Linear Search	Binary Search	Ternary Search	Deteksi	Eksploitasi	Web online	Local	Runtime	*Request*
Study the Best PenTest Algorithm for Blind SQL Injection Attacks. (Nugroho & Mandala, 2020)		✓	✓	✓			✓		✓	✓	
Simulating SQL Injection Vulnerability Exploitation Using Q-Learning Reinforcement Learning Agen. (Erdodi dkk., 2021)	✓	✓					✓				
An algorithm for detecting SQL injection vulnerability using black-box testing. (Aliero dkk., 2020)		✓				✓		✓			
Time-Based Blind SQL Injection via HTTP Headers: Fuzzing and Exploitation. (Ombagi, 2017)	✓						✓		✓		
Complexity Analysis between Bruteforcing and Blind SQL Injection with LIKE-Based Data Exfiltration. (Jibril Ibrahim -, 2024)	✓	✓					✓				
Optimizing Search Efficiency in Ordered Data: A Hybrid Approach				✓	✓				✓	✓	

Judul Penelitian	Ruang Lingkup Penelitian										
	Blind SQL i		Algoritma			Tujuan		Objek		Parameter Evaluasi	
	Time Blind SQLi	Boolean Blind SQLi	Linear Search	Binary Search	Ternary Search	Deteksi	Eksploitasi	Web online	Local	Runtime	*Request*
Using Jump Binary Search. (Rorong dkk., 2025)											
Analisis Perbandingan Algoritma *Ternary* dan *jump search.* (Oleh & Dwiyanto, 2020.)					✓				✓	✓	
The ART of SQL Injection Vulnerability Discovery. (Zhang dkk., 2019)		✓				✓					
PENETRATION TESTING PROCESS FOR SQL INJECTION ATTACK. (Hajar Ismail dkk., 2024)	✓	✓				✓					
SQL Injection: Analisis Efektivitas Uji Penetrasi dalam Aplikasi Web. (Nugraha dkk., 2024)		✓				✓		✓			
Our Research	✓		✓	✓	✓		✓	✓		✓	✓

Berdasarkan Tabel 2.3, terlihat bahwa sebagian besar penelitian terdahulu masih memiliki keterbatasan, baik dari sisi pendekatan algoritma maupun parameter evaluasi yang digunakan. Beberapa studi seperti yang dilakukan oleh Alanda dkk. (2023), Aliero dkk. (2020), dan Zhang dkk. (2019) lebih berfokus pada aspek deteksi kerentanan dan belum menyentuh secara teknis proses eksploitasi terhadap data *database*. Di sisi lain, penelitian yang sudah mengarah pada eksploitasi seperti yang dilakukan oleh Pružinec & Anh Quynh (2023), Nugroho & Mandala (2020), serta Firnando & Sijabat (2024), umumnya hanya memanfaatkan sebagian algoritma pencarian seperti *binary search* atau *linear search* secara terpisah, tanpa pembandingan menyeluruh. Penelitian ini menempati posisi yang lebih komprehensif karena membandingkan tiga algoritma pencarian (*linear, binary, dan ternary search*), menguji dan membandingkan eksploitasi otomatis dengan algoritma *Linear*,*Binary* dan *Ternary search* terhadap *Time-Based Blind SQL Injection* pada aplikasi web nyata, serta mengevaluasi parameter *runtime*, total *request*, dan akurasi hasil ekstraksi dari masing masing algoritma, disertai rekomendasi mitigasi sebagai kontribusi tambahan.