BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Dalam penelitian ini akan digunakan lima pustaka hasil-hasil penelitian terdahulu yang relevan. Adapun penelitian terdahulu yang diambil sebagai referensi adalah :

1. (Nugroho & Mandala, 2020) dari Fakultas Informatika, Universitas Telkom dengan judul *Study the Best PenTest Algorithm for Blind SQL Injection Attacks*. Pada penelitian ini telah dilakukan pengukuran antara 3 algoritma linear, binary dan interpolasi untuk mencari algoritma yang paling tepat digunakan dalam serangan SQL *Injection* jenis *Blind* pada DBMS SQL dengan metode penelitian yang digunakan kualitatif. Algoritma *Binary Search* merupakan algoritma tercepat berdasarkan hasil pengujian dengan waktu 1,7852 detik, kemudian *Interpolation Search* dengan 1.789 detik dan *Linear Search* 1.902 detik.
2. (Singh, 2019), Carnegie Mellon University dengan judul *Security Analysis of MongoDB*. Pada penelitian ini telah dilakukan analisa keamanan dasar dan kerentanan MongoDB serta bagaimana kinerjanya dibandingkan dengan *database* relasional. Dengan metode penelitian yang digunakan kualitatif. Pengujian dilakukan dengan aspek s*ecurity* model, fitur dan kesesuaian NoSQL terhadap kemampuannya menghadapi Big data yang tidak terstruktur. Pada makalah ini juga terdapat pembahasan yang mendalam tentang bagaimana sebuah serangan *NoSQL Injection*

terjadi. Penelitian ini juga memuat kasus insiden yang pernah terjadi pada sejumlah aplikasi yang menggunakan DBMS NoSQL MongoDB.

3. (Guptaa, Singha, & Tomara, 2018) dari Fakultas Ilmu Komputer, Maulana Azad National Institute of Technology dengan judul *Analysis of NoSQL Database Vulnerabilities*. Pada Penelitian ini telah didiskusikan tentang berbagai macam kerentanan dan jenis serangan pada NoSQL seperti metode *bypass* otentikasi, *Javascript file inclusion, Blind NoSQL Injection* dan *DoS.* Pada makalah ini juga dibahas macam-macam *attack vector.* Hasil dari penelitian ini *database* NoSQL memiliki banyak kelebihan dan kegunaan pada organisasi besar karena banyak fiturnya. Namun disamping itu NoSQL juga memiliki banyak masalah pada bagian keamanan dan selalu saja ada celah keamanan yang dapat dieksploitasi oleh para penyerang dan masalah keamanan ini harus dapat ditangani dan dibutuhkan lebih banyak solusi deteksi dan teknik untuk mempertahankan dari serangan sehingga lingkungan yang aman dapat dibangun untuk menggunakan *database* NoSQL.

4. (Priyadharshini & Rajmohan, 2017) dari fakultas Ilmu komputer, Anna University/IFET College of Engineering dengan judul *Analysis on Database Security Model Against NoSQL Injection*. Pada penelitian ini dibahas tentang kematangan dari langkah-langkah keamanan untuk MongoDB dan ciri khas sistem *database* NoSQL dengan aspek serangan dan pertahanan pada level kode. Pada penelitian ini MongoDB dan Cassandra digunakan sebagai objek uji coba.
5. (Hou, Shi, Qian, & Tao, 2017) dari Fakultas Ilmu Komputer, Kennesaw State University dan Pace University dengan judul *Towards Analyzing MongoDB NoSQL Security and Designing Injection Defense Solution*. Pada penelitian ini telah dibahas mengenai serangan *NoSQL Injection* pada MongoDB serta diusulkan cara pendeteksian dan beberapa pendekatan untuk bertahan dari serangan jenis injeksi. Ada 3 jenis pendekatan sebagai solusi pertahanan dari serangan pada NoSQL, yang pertama adalah validasi input dari user dengan cara menambahkan regex (*reguler expression)*. Solusi kedua, dengan menetapkan hak akses kepada semua user sehingga dapat mencegah terjadinya *JS Injection.* Solusi ketiga dengan cara mengecek dan menyaring *variable* yang berisi sebuah *statement.* Hasil dari penelitian ini menunjukkan bahwa satu tindakan pengamanan tidak dapat menjamin keamanan komunikasi dan informasi. Dengan begitu harus diintegrasikan dengan berbagai tindakan pengamanan melalui sarana teknis dan administratif.

6. (Shachi et al., 2021) dari Fakultas Ilmu Komputer, Ahsanullah University of Science and Technology dengan judul *A Survey on Detection and Prevention of SQL and NoSQL Injection Attack on Server-side Applications*. Pada penelitian telah dibahas tentang ancaman serangan injeksi SQL dan NoSQL pada aplikasi web, serta upaya untuk mendeteksi serangan dan menerapkan tindakan pencegahan guna mengurangi kerentanan terhadap serangan injeksi tersebut. Jika database dapat diinjeksi dengan *malicious code*, data rahasia dapat terbocorkan kepada peretas. SQL *injection* merupakan ancaman serius terhadap *database* karena validasi input pengguna yang kurang memadai. Baik *database* relasional maupun non-relasional rentan terhadap serangan ini. Basis data NoSQL menunjukkan kinerja lebih tinggi daripada basis data SQL dalam hal penyimpanan efisien dan waktu pengambilan data. Meskipun demikian, basis data NoSQL tetap rentan terhadap serangan injeksi.

7. (Fitri, 2013) dari Jurusan Teknik Informatika Fakultas Sains dan Teknologi, Universitas Islam Negeri Alauddin Makassar dengan judul TREND PENGGUNAAN NOSQL UNTUK BASIS DATA NON RELASIONAL. Pada penelitian ini telah dibahas tentang NoSQL sebagai sebuah sistem basis data non-SQL yang berasal dari sistem RDBMS. NoSQL menggunakan sistem operasi UNIX. Paper ini menjelaskan tentang NoSQL dan bagaimana melakukan pemodelan basis data dengan NoSQL. NoSQL memiliki beberapa keunggulan seperti skalabilitas yang elastis, kemampuan mengelola big data, penghapusan DBA (*Database*

Administrator), efisiensi ekonomi, dan model data yang fleksibel. Namun, NoSQL juga memiliki beberapa tantangan seperti kematangan, dukungan, analitik dan *business intelligence*, administrasi, dan keahlian.

8. (Joe & Selvarajah, 2022) dari Asia Pacific University of Technology & Innovation (APU), Malaysia dengan judul *A Study of SQL Injection Hacking Techniques*. Pada penelitian ini telah dibahas pengkategorian serangan SQL *injection* berdasarkan karakteristiknya dan menyoroti bahwa tingkat keparahan serangan dapat bervariasi, mulai dari kebocoran data hingga penghancuran dan modifikasi besar-besaran pada *database*. Makalah ini memberikan gambaran umum tentang serangan SQL *injection*, melakukan demonstrasi serangan pada sebuah database, dan membahas karakteristik serta contoh pemanfaatan kerentanan SQL *injection* yang berbeda.
9. (Focardi, Luccio, & Squarcina, 2012) dari Universit`a Ca' Foscari Venezia dengan judul *Fast SQL blind injections in high latency networks*. Secara khusus, penelitian ini membahas tentang *Blind* SQL *Injection*, di mana serangan ini tidak mengembalikan data secara langsung, tetapi memberikan informasi 1-bit tentang keberhasilan *query*. Biasanya, mendapatkan seluruh database melalui *Blind* SQL *Injection* membutuhkan waktu yang cukup lama, terutama dalam jaringan dengan latensi tinggi, yang meningkatkan kemungkinan terdeteksinya serangan oleh administrator sistem. Untuk meningkatkan efektivitas dan kecepatan serangan *Blind* SQL *Injection*, penelitian ini memperkenalkan perbaikan

seperti pencarian berbasis probabilitas dan berbasis kamus, serta paralelisasi *query*. Penelitian ini menunjukkan bahwa perbaikan-perbaikan ini secara signifikan meningkatkan efisiensi dan efektivitas serangan, bahkan dalam lingkungan jaringan dengan latensi tinggi.

10. (Subbarayudu, et al., 2018) dari Dept. of ECE, Institute of Aeronautical Engineering dengan judul *COMPARATIVE ANALYSIS ONSORTING AND SEARCHING ALGORITHMS.* Pada penelitian ini membahas tentang berbagai algoritma pengurutan dan pencarian. Analisis ini mengungkapkan kelebihan dan kekurangan dari berbagai algoritma pengurutan dan pencarian beserta contohnya. Berbagai teknik pengurutan dianalisis berdasarkan kompleksitas waktu dan kompleksitas ruang. Melalui analisis ini, ditemukan bahwa quick sort efektif untuk data dalam jumlah besar dan insertion sort efisien untuk daftar kecil. Selain itu, ditemukan bahwa binary search cocok untuk data dalam jumlah sedang dan dapat diterapkan pada array dan linked list, sedangkan hash search merupakan yang terbaik untuk data dalam jumlah besar, dan exponential search dapat digunakan untuk himpunan elemen yang tak terbatas.
11. (Macnish & van der Ham, 2020) dari University of Twente dengan judul *Ethics in cybersecurity research and practice.* Penelitian ini membahas tentang tata kelola etika dalam konteks keamanan siber. Tujuannya adalah untuk memberikan gambaran tentang masalah-masalah etika yang dihadapi oleh peneliti dalam komunitas keamanan siber, baik dari kalangan akademik maupun praktisi di perusahaan. Penelitian ini

mengidentifikasi dua kelompok masalah etika: yang dihadapi oleh komunitas penelitian akademik dan praktisi. Dua studi kasus digunakan untuk mengilustrasikan perbedaan antara kedua kelompok ini. Penelitian ini menekankan perbedaan dalam pengawasan etika antara kedua kelompok tersebut. Para peneliti akademik memiliki badan etika penelitian yang dapat memberikan pengawasan dan tata kelola etika. Namun, penelitian ini juga mencatat bahwa dalam beberapa kasus, badan etika tersebut mungkin tidak memberikan nasihat yang memadai. Rekomendasi dari penelitian ini adalah perlunya pemahaman yang lebih mendalam tentang etika dalam kurikulum ilmu komputer, serta implementasi kode etik yang efektif. Selain itu, pentingnya diskusi mengenai etika dalam keamanan siber dan penelitian keamanan siber sebagai upaya untuk mengatasi tantangan dalam tata kelola etika di bidang tersebut.

Berdasarkan penelitian terdahulu tersebut, akan dikembangkan suatu *tool* yang akan berfokus pada MongoDB yang berbasis NoSQL yang dirancang untuk memudahkan pengujian keamanan terhadap serangan *blind injection.*

2.2 Teori Terkait

2.2.1 NoSQL

NoSQL merupakan suatu metode penyimpanan data (datastore) yang memungkinkan penggunaannya dalam operasi penyimpanan dan pengambilan data yang cepat, serupa dengan basis data relasional. Namun, yang membedakannya adalah NoSQL tidak mengandalkan hubungan matematis antar tabel seperti yang terdapat pada basis data relasional (Fitri, 2013). NoSQL juga salah satu jenis *database* yang terdistribusi, yang berarti informasi disalin dan disimpan pada berbagai server, yang mana dapat diakses secara *remote* atau lokal (IBM, n.d.).

NoSQL menyediakan beberapa opsi untuk mengorganisir data dengan beberapa cara. Dengan menawarkan struktur data yang beragam, NoSQL dapat diterapkan pada analitik data, mengelola *big-data*, jejaring sosial, dan pengembangan aplikasi *mobile*. *Database* NoSQL umumnya dikategorikan menjadi 4 model penyimpanan data:

1. *Key-value database* : Jenis database yang menggunakan struktur data sederhana berupa pasangan kunci (*key*) dan nilai (*value*). Setiap item dalam *database* diidentifikasi oleh kunci unik dan nilainya dapat berupa tipe data apapun. Dalam *key-value database*, data disimpan dalam tabel *hash*.

2. *Column database* : Jenis *database* yang menyimpan data dalam bentuk kolom atau kolom kelompok, bukan dalam bentuk baris seperti pada *database* relasional. Kunci dapat berupa penanda waktu yang menunjukkan satu atau beberapa kolom (*Column Family*). Setiap baris dalam *column database* tidak harus memiliki kolom yang sama, sehingga memberikan fleksibilitas dalam struktur data.

3. *Document database* : Jenis *database* yang menyimpan data dalam dokumen, yang biasanya mengadopsi format JSON (*JavaScript Object Notation*) atau XML. Setiap dokumen dapat memiliki struktur yang berbeda dan data dapat diatur dalam hierarki yang kompleks. Konsep dari *document database* ini lebih efisien dan fleksibel.

4. *Graph Database* : Jenis *database* NoSQL yang dirancang khusus untuk menyimpan dan mengelola hubungan antara entitas. Data dalam graph database disimpan dalam bentuk *node* dan *edge*. *Node* mewakili entitas seperti orang, tempat, atau objek lainnya, sedangkan *edge* menggambarkan hubungan antara *node-node*

tersebut. *Graph database* sangat efektif dalam menganalisis dan menavigasi struktur data yang kompleks. (Guptaa, Singha, & Tomara, 2018)

2.2.2 MongoDB *NoSQL Injection*

MongoDB merupakan salah satu produk *database* non-relasional yang banyak digunakan dimana sistem ini menggunakan struktur data JSON untuk proses tersimpannya data. MongoDB menyimpan data dalam bentuk encode JSON yang biasa disebut dengan BSON. BSON ini cukup sesuai di era modern object oriented programming. Database non-relasional MongoDB menggunakan sintaks BSON yang cukup berbeda dibandingkan dengan MySQL. Namun, kajian ini menyimpulkan bahwa MongoDB masih memiliki kelemahan dalam mencegah serangan injeksi. Serangan injeksi pada DBMS NoSQL biasa disebut dengan *NoSQL Injection* (Shachi et al., 2021).

Injeksi adalah suatu metode serangan yang umum digunakan untuk mengeksploitasi *database*. Serangan injeksi pada DBMS NoSQL biasa disebut dengan *NoSQL Injection*. Serangan injeksi *database* ini dapat dilakukan dengan menyisipkan *malicious code (*kode jahat) untuk mengeksekusi kode dari aplikasi atau *software.* Dengan memanfaatkan celah ini *Penetration Tester* dapat memperoleh beberapa otoritas sebagai user atau root maupun memperoleh informasi *database*, memperbarui atau menghapus data pada *database*. Pada dasarnya, *Penetration Tester* pertama-tama harus menemukan letak *inject point* untuk menyisipkan kode jahat agar dapat dieksekusi oleh aplikasi. Kedua,

Penetration Tester harus mengetahui jenis *database* yang digunakan korban. Ketiga, *malicious code* disusun untuk menginjeksi dan eksfiltrasi data pada *database*. Pada dasarnya, serangan pada NoSQL mirip dengan SQL. *Penetration Tester* memasukkan *malicious code* pada *form input* atau langsung pada URL. Secara umum, *malicious code* seringkali dijalankan pada proses pencarian, dalam kondisi normal program akan mencari baris demi baris dan menampilkan hasil. Ketika *malicious code* disuntikkan dalam proses pencarian, hasil kalimat itu akan menjadi statement yang selalu benar ataupun dapat mengontrol data apa yang ingin diekstrak dari *database*, sehingga program akan menampilkan sesuai apa yang diinginkan *Penetration Tester*.

2.2.3 Blind *NoSQL Injection*

Serangan jenis *blind* pada *NoSQL Injection* mirip dengan serangan *Blind* pada *SQL Injection*. Data tidak muncul pada halaman web karena pada suatu kondisi tertentu sebuah *inject point* terdapat di suatu fitur yang tidak menampilkan *output* data. *Penetration Tester* masih dapat menyimpulkan dari *payload* yang dikirimkan ke *server* dengan cara memanipulasi *query* dan melakukan pengamatan pada hasil respon yang diberikan.

Umumnya, ada dua jenis serangan jenis *Blind NoSQL Injection*, yaitu:

Boolean-based *:* Boolean-based adalah jenis serangan *Blind NoSQL Injection* dimana sebuah website hanya menampilkan output *True* dan *False* setelah *payload query* di-*request* ke dalam server.

Time-based : *Time-based* adalah jenis serangan *Blind NoSQL Injection* yang mana serangan ini mengandalkan sebuah teknik waktu *delay database* sebelum memberikan respon, sehingga seorang *Penetration Tester* dapat memanfaatkan ini untuk mendapatkan kesimpulan dari hasil *query* yang dikirimkan menghasilkan respon *True* atau *False*

2.2.4 Python

Python adalah sebuah bahasa pemrograman open source yang termasuk dalam kategori bahasa pemrograman tingkat tinggi. Bahasa ini memiliki sifat interpreted, interaktif, dan berorientasi objek. *(What is Python? Executive Summary, n.d.)*. Python diklaim sebagai bahasa yang menggunakan gabungan *modules,exceptions, dynamic typing*, *high level dynamic data types* dan banyak *class.* Bahasa Python juga mendukung beberapa paradigma pemrograman seperti OOP, prosedural, imperatif dan fungsional. Python diklaim memiliki pustaka standar yang besar serta komprehensif.

Beberapa modul yang sering digunakan untuk membuat *tool* untuk kebutuhan pengujian keamanan adalah sebagai berikut:

1. *Requests*

   Requests adalah sebuah modul Python yang digunakan untuk melakukan *request* HTTP. Modul ini menyediakan berbagai fitur yang berguna dalam request HTTP, seperti kemampuan untuk mengirimkan *parameter* melalui URL, mengirimkan *header* kustom, melakukan

verifikasi SSL, serta menyederhanakan proses *request* dan *receive* respons HTTP.

2. *BeautifulSoup*

   BeautifulSoup adalah sebuah *library* Python yang populer dan sering digunakan untuk melakukan *parsing* atau ekstraksi data dari elemen HTML. Dengan menggunakan *BeautifulSoup*, pengguna dapat dengan mudah mengurai struktur HTML dan mengekstrak informasi yang diinginkan, seperti teks, atribut, atau elemen tertentu dari halaman *web*.

3. Pwntool

   Pwntool adalah sebuah *library* python yang digunakan untuk keperluan *exploit development.* Terdapat banyak fitur pada *library* pwntools yang tersedia sehingga mempermudah proses pengembangan *tool exploit*. Fitur yang terkenal dari *library* ini adalah *remote exploit* berbasis *socket.*

4. String

   String merupakan *library* bawaan python yang menyediakan banyak konstanta, fungsi utilitas dan beberapa *class* untuk memanipulasi data string.

2.2.5 NodeJS

Node.js merupakan bahasa pemrograman *open source* yang dibangun menggunakan runtime bahasa JavaScript sehingga pengguna dapat menuliskan bahasa JavaScript diluar *browser*. NodeJS dibangun untuk membangun jaringan

aplikasi yang skalabel. Node.js menggunakan JavaScript sebagai bahasa pemrograman dan mengadopsi model I/O (*input/output*) *non-blocking* (*asynchronous*) yang berbasis *event*, sehingga membuatnya menjadi *platform* yang ringan dan efisien. (NodeJS, n.d.).

Pada dasarnya, Node.js dikembangkan menggunakan JavaScript dan C++. Node.js menggunakan arsitektur serta fungsi dari *V8 Engine* yang berperan sebagai *compiler* yang ditulis dalam bahasa C++, serta *library Libuv* yang digunakan untuk menghandle operasi *I/O asynchronous* dan *main event loop*. (V8, n.d.).

Pada penelitian ini, digunakan *framework* ExpressJS sebagai basis pengembangan lab simulasi karena kemampuannya untuk menghandle *server-side* cukup fleksibel.

Beberapa modul yang akan digunakan pada penelitian ini, sebagai berikut:

a. Body-parser

   body-parser merupakan modul pada Node.js digunakan pada *framework* Express untuk mengekstrak data dari form.

b. Cors

   CORS adalah modul yang mengizinkan sebuah request untuk mengirimkan method POST, GET, PUT, dan DELETE ke server.

c. Mongoose

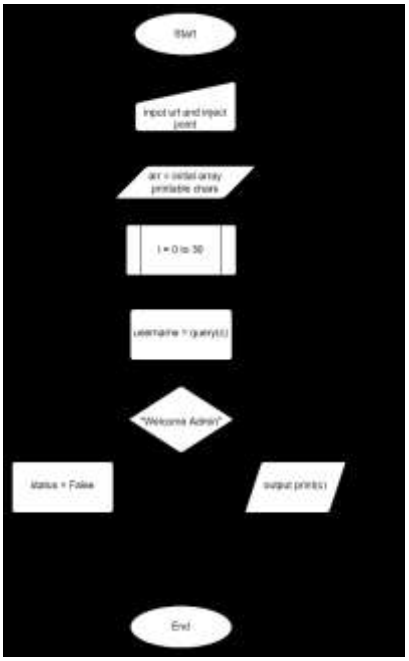
Mongoose merupakan modul NodeJS yang digunakan sebagai penghubung aplikasi dengan *database* MongoDB yang diinstall melalui NPM (*Node Package Manager*).

2.2.6 Pencarian Berurutan (*Sequential Searching*)

Metode pencarian yang paling sederhana disebut sebagai pencarian berurutan atau pencarian *linear*. Dalam pencarian berurutan, data dibandingkan satu per satu secara berurutan dengan data yang sedang dicari. Proses ini berlangsung sampai data tersebut ditemukan atau tidak ditemukan.

Pada dasarnya, metode pencarian ini melibatkan pengulangan dari 1 hingga jumlah data. Pada setiap pengulangan, data ke-i dibandingkan dengan data yang sedang dicari. Jika data tersebut sama, berarti data telah ditemukan. Namun, jika tidak ada kesamaan pada akhir pengulangan, berarti data tidak ada. Dalam kasus terburuk, pencarian ini akan dilakukan sebanyak N kali untuk N elemen data.

Berikut merupakan gambaran proses injeksi menggunakan Algoritma *Linear Search* yang akan dibangun. Dapat dilihat pada gambar berikut.



Gambar 2.1 Flowchart Algoritma *Linear Search*

2.2.7 Pencarian *Binary* (*Binary Search*)

Binary Search adalah metode untuk mencari data dalam sebuah *array* yang telah terurut. Salah satu persyaratan utama dalam *Binary Search* adalah data harus dalam keadaan terurut. Dengan kata lain, jika data belum terurut, maka *Binary Search* tidak dapat dilakukan. Tujuan dari *Binary Search* ini adalah sebagai berikut:

a. Mengurangi jumlah operasi perbandingan yang perlu dilakukan antara data yang dicari dengan data yang ada dalam tabel, terutama saat jumlah data sangat besar.
b. Beban komputasi dapat diperkecil dengan melakukan pencarian dari depan, belakang, dan tengah.

c. Prinsip dasarnya adalah melakukan pembagian ruang pencarian secara berulang-ulang hingga data ditemukan atau hingga pencarian tidak dapat dibagi lagi, yang menunjukkan kemungkinan data tidak ditemukan.

Untuk melakukan *binary search*, data harus dalam keadaan terurut. Artinya, jika data tidak terurut, *binary search* tidak dapat dilakukan. Prinsip dasar dari *binary search* adalah sebagai berikut:

1. Tentukan posisi awal sebagai 0 dan posisi akhir sebagai N - 1, di mana N adalah jumlah total elemen dalam data..
2. Hitung posisi tengah dengan menggunakan rumus (posisi awal + posisi akhir) / 2.
3. Bandingkan data yang dicari dengan data pada posisi tengah. Jika data yang dicari lebih kecil, ulangi proses pencarian dengan memperbarui posisi akhir menjadi posisi tengah - 1. Jika data yang dicari lebih besar, ulangi proses pencarian dengan memperbarui posisi awal menjadi posisi tengah + 1.
4. Demikian seterusnya sampai data pada posisi tengah sama dengan data yang dicari.

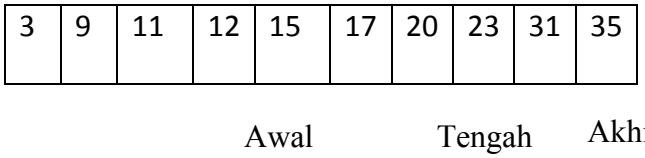
Untuk lebih jelasnya perhatikan contoh berikut:

Misalnya ingin mencari data 17 pada sekumpulan data berikut.

3	9	11	12	15	17	20	23	31	35

Awal Tengah Akhir

Langkah pertama dalam pencarian biner adalah mencari data tengah. Dalam contoh ini, data tengah berada di posisi ke-4 dengan nilai 15, dihitung menggunakan rumus (posisi awal + posisi akhir) / 2 = (0 + 9) / 2 = 4. Selanjutnya, data yang dicari (17) dibandingkan dengan data tengah ini. Karena 17 lebih besar dari 15, proses dilanjutkan dengan menganggap posisi awal baru sebagai posisi tengah + 1, yaitu 5.

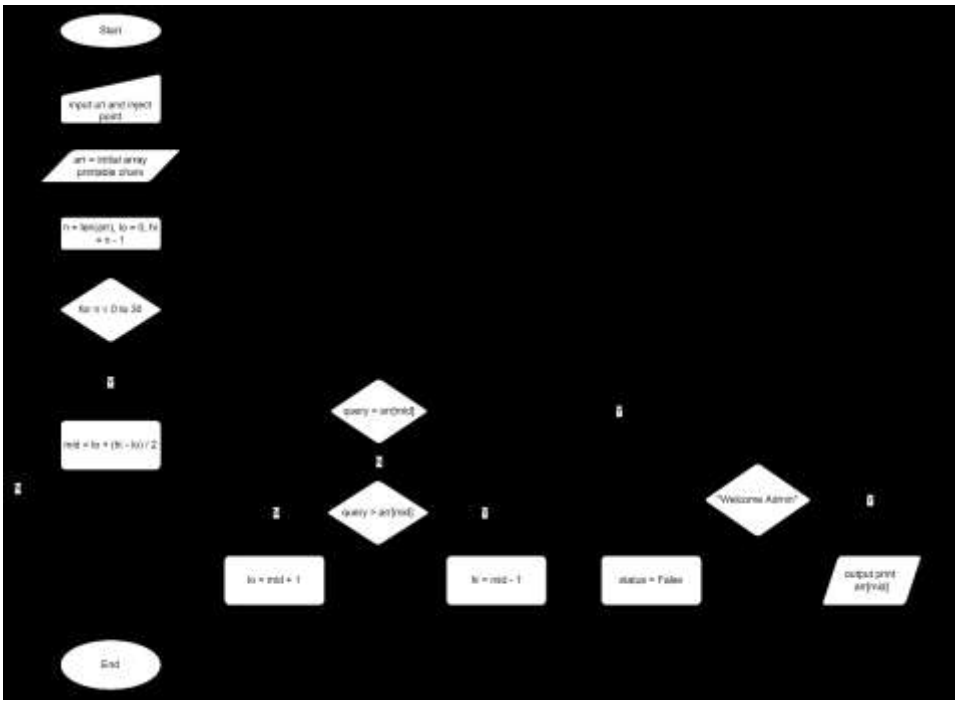


Data tengah yang baru dihitung dengan rumus (posisi awal + posisi akhir) / 2 = (5 + 9) / 2 = 7. Dengan demikian, data tengah yang baru berada pada posisi ke-7 dengan nilai 23. Selanjutnya, data yang dicari (17) dibandingkan dengan data tengah ini. Karena 17 lebih kecil dari 23, proses dilanjutkan dengan menganggap posisi akhir baru sebagai posisi tengah -1 atau 6.

Data tengah yang baru diperoleh dengan mengaplikasikan rumus (5 + 6) / 2 = 5. Oleh karena itu, data tengah yang baru adalah data ke-5, yaitu 17. Data yang dicari dibandingkan dengan data tengah ini, dan ternyata kedua data tersebut sama. Sehingga, data ditemukan pada indeks ke-5. Proses pencarian biner ini akan

berakhir jika data ditemukan atau jika posisi awal melebihi posisi akhir. Jika posisi awal sudah melebihi posisi akhir, berarti data tidak ditemukan. Jumlah pembandingan minimal dalam pencarian biner adalah 1 kali, terjadi ketika data yang dicari tepat berada di posisi tengah. Jumlah pembandingan maksimum dalam pencarian biner dapat dihitung menggunakan rumus logaritma, yaitu: C = 2 log v (N).

Berikut merupakan gambaran proses injeksi menggunakan Algoritma *Binary Search* yang akan dibangun. dapat dilihat pada gambar berikut.



Gambar 2.2 Flowchart *Algoritma Binary Search*

2.2.8 *Penetration Tester*

Seorang *penetration tester* (pentester) atau *ethical hacker* adalah seorang profesional keamanan komputer yang secara etis mengidentifikasi dan mengeksplorasi celah keamanan dalam sistem komputer, jaringan, aplikasi, atau infrastruktur IT. Tujuan utama dari seorang *penetration tester* adalah untuk menemukan kerentanannya sebelum penyerang yang jahat dapat mengeksploitasi mereka. *Penetration tester* menggunakan metode dan alat yang mirip dengan apa yang digunakan oleh penyerang sungguhan, tetapi dengan izin dan tujuan yang jelas. (Weidman, 2014)