BAB III
ANALISIS DAN PERANCANGAN

3.1 Analisis

3.1.1 Identifikasi Masalah

Identifikasi masalah merupakan tahap awal untuk mengetahui suatu permasalahan yang akan diteliti. Pengujian keamanan perangkat lunak sering digunakan untuk menemukan kerentanan keamanan dalam perangkat lunak atau aplikasi. Keamanan sebuah perangkat lunak juga berpengaruh pada privasi atau kerahasiaan pengguna perangkat lunak atau aplikasi. Berdasarkan latar belakang yang telah dijelaskan sebelumnya, saat ini sudah ada *tool* yang ada di internet untuk membantu *Penetration Tester* dalam pengujian NoSQL Injection yaitu NoSQLMap. Kekurangan dari NoSQLMap ini hanya menerapkan Algoritma *Linear Search* dan berjalan secara *single-thread* sehingga cukup lama dan kurang efektif apabila dilakukan pada jenis serangan *Blind* NoSQLi. Karena MongoDB tidak menggunakan *query* SQL bukan berarti MongoDB kebal terhadap serangan injeksi. Injeksi masih mungkin terjadi karena MongoDB mengijinkan *arbitrary JavaScript expression* untuk dieksekusi langsung di *server.*

SQL Injection merupakan teknik serangan injeksi pada *database* berbasis SQL. Berikut salah satu contoh serangan pada *classic* SQL *Injection.*

*SELECT * FROM users WHERE username = '' or 1=1-- -' AND password = 'x'*

Query diatas menunjukkan bagaimana SQL *Injection* terjadi karena 1=1 selalu bernilai benar, *query* diatas dieksekusi dan penyerang mungkin bisa mendapatkan akses ke informasi pribadi (misalnya kredensial). Oleh karena itu, pernyataan seperti itu mungkin tanpa sadar banyak terjadi.

Database NoSQL menggunakan bahasa *query* yang berbeda dengan SQL yang membuat injeksi SQL tradisional tidak relevan untuk *database* NoSQL. Namun, *database* NoSQL tidak kebal dari serangan injeksi dan penyerang dapat berkreasi dengan teknik fundamental yang telah di*improve* untuk ini. *Query* yang *equivalent* dengan database SQL yang dibahas di atas pada database NoSQL MongoDB adalah:

```
db.users.find({
   "$where": function() {
      (this.username == || 1==1%00 ) && (this.password == 'x')
   }}
)
```

Dalam hal ini, penyerang masih dapat menyisipkan *query* ini menggunakan objek JSON (BSON) sebagai berikut:

```
{
      "username": "x",
      "password": {$ne: NULL}
}
```

Kedua statement di atas menghasil kan nilai *True*, Pada MongoDB $ne akan memeriksa dan menyeleksi dokumen password yang nilainya tidak sama dengan *NULL* sehingga statement akan bernilai *True*.

Dalam penelitian ini *user* perlu menginputkan *inject point* dari *parameter* yang *vulnerable* terhadap serangan kemudian memilih jenis dan mode eksploitasi. Pada aplikasi *user* dapat memilih beberapa algoritma yang dapat digunakan menggunakan Algoritma *Linear* atau *Binary Search.*

Algoritma *Linear* membutuhkan waktu yang cukup lama dalam proses eksfiltrasi data pada serangan *Blind*. Hal ini disebabkan oleh pendekatan pencarian berurutan yang dilakukan oleh Algoritma *Linear*, dimana pencarian dilakukan dengan mengulangi setiap elemen data dari awal hingga akhir. Berbeda dengan Algoritma *Binary Search*, yang menggunakan pendekatan membagi dua dan melakukan tebakan data yang dicari berada di bagian tengah atau tidak, dengan syarat bahwa data dalam array sudah terurut sebelumnya.

Pada penelitian terdahulu oleh Aldebaran Bayu Nugroho dengan judul "*Study the Best PenTest Algorithm for Blind SQL Injection Attacks"* dengan DBMS MySQL yang berbasis SQL sebagai objek penelitian didapatkan kesimpulan *Binary Search* lebih cepat dibandingkan dengan *Linear Search.* Berbeda dengan DBMS NoSQL, DBMS NoSQL saat ini masih belum diketahui lebih baik algoritma *Binary Search* atau *Linear Search* yang efisien untuk serangan NoSQL *Injection* berjenis *blind.* Dalam penelitian ini akan dilakukan perbandingan kinerja kedua algoritma tersebut. Berdasarkan perbandingan, pada

Tugas Akhir ini akan dikembangkan suatu *tool* untuk NoSQL *Injection* dengan menerapkan Algoritma *Binary Search.*

3.1.2 Pemecahan Masalah

Dari hasil analisa permasalahan, timbul kebutuhan untuk menyelesaikan semua masalah yang dihadapi. Untuk mencapai tujuan tersebut, diperlukan sebuah tool exploit yang dapat memanfaatkan Algoritma Linear Search dan Binary Search dalam menguji kerentanan Blind NoSQL Injection.

Dengan penerapan Algoritma *Binary Search* pada *tool expoit Blind* NoSQLi dapat mengoptimalkan pengujian celah keamanan NoSQL berjenis *Blind* sehingga proses injeksi dapat dilakukan lebih efektif serta mempercepat proses pengujian keamanan oleh *Penetration Tester*. Pemilihan bahasa pemrograman Python sebagai dasar pengembangan didasarkan pada beberapa faktor. Pertama, Python adalah bahasa pemrograman yang populer dan memiliki berbagai *library* yang mendukung pengembangan aplikasi keamanan. Kedua, Python memiliki sintaks yang sederhana dan mudah dipahami. Ketiga, dukungan yang luas dalam komunitas Python memudahkan dalam memperoleh bantuan dan sumber daya terkait pemecahan masalah yang mungkin timbul selama pengembangan.

3.2 Perancangan

3.2.1 Perancangan Sistem

Setelah melakukan analisis terhadap masalah yang ada, langkah berikutnya adalah merancang sistem. Perancangan sistem ini memiliki tujuan untuk memberikan gambaran yang lebih terperinci mengenai fungsionalitas sistem yang

akan dibuat. Diharapkan bahwa perancangan ini akan memberikan kontribusi yang signifikan dalam menyelesaikan masalah yang dihadapi. Perancangan sistem ini akan mencakup beberapa aspek penting, seperti desain data yang merinci struktur dan hubungan antara entitas, desain aplikasi yang menjelaskan logika dan alur kerja sistem, serta desain antarmuka yang menentukan tampilan dan interaksi antara pengguna dengan sistem.

3.2.2 Perancangan Laboratorium Simulasi

Perancangan laboratorium simulasi sebagai sarana untuk melakukan uji coba serangan MongoDB Injection. Laboratorium simulasi ini dibangun menggunakan NodeJS dan MongoDB sebagai basis datanya. Laboratorium simulasi ini bertujuan untuk memberikan pemahaman dan pengalaman dalam simulasi serangan MongoDB Injection dengan penerapan Algoritma *Linear* dan optimasi dengan *Binary Search*, serta memberikan solusi untuk mencegah dan mengatasi serangan tersebut. Laboratorium simulasi ini dirancang dengan menggunakan paradigma MVC.

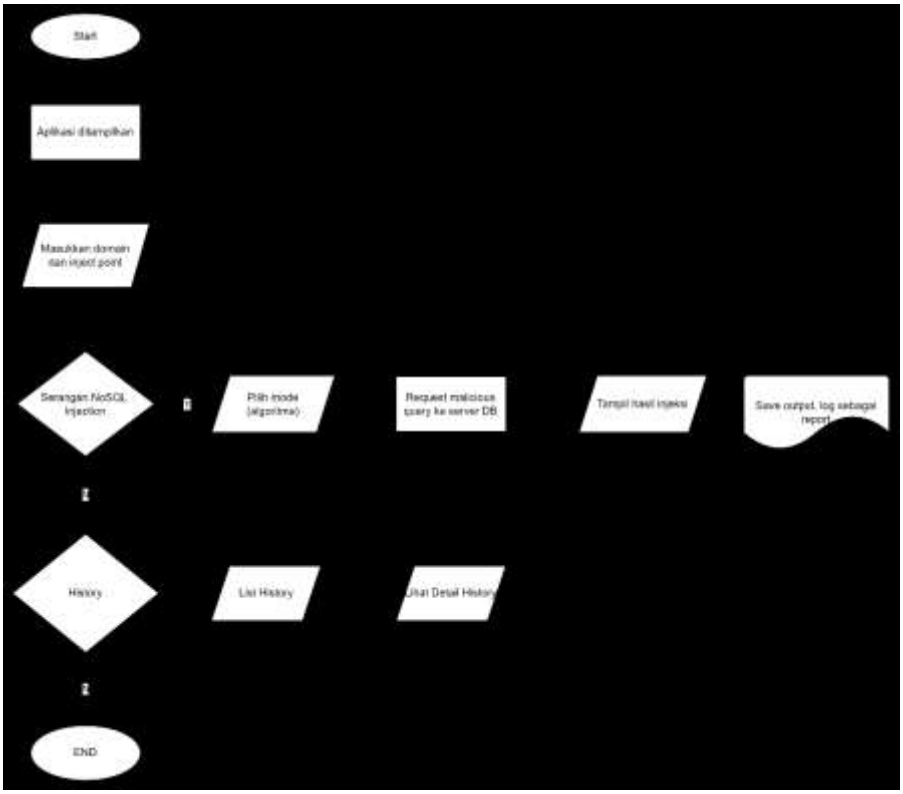
3.2.3 Perancangan Flowchart

Perancangan diagram alir ini menjadi salah satu hal yang penting dalam pembuatan sebuah sistem. Diagram alir, juga dikenal sebagai flowchart, digunakan untuk menjelaskan bagaimana alur program dijelaskan melalui berbagai bagan yang memiliki fungsi yang berbeda, seperti bagan input/output dan bagan proses. Bagan-bagan ini dihubungkan oleh garis panah yang menunjukkan urutan jalannya sistem.

3.2.2.1 Flowchart Sistem

Flowchart ini menggambarkan bagaimana sistem aplikasi beroperasi. Tujuannya adalah untuk menguraikan proses secara sistematis yang menjelaskan aktivitas-aktivitas yang terjadi dalam aplikasi yang sedang dibangun. Aplikasi dapat melakukan eksfiltrasi data pada *database* dengan dua pilihan algoritma yakni *Linear* dan *Binary Search.*

Berikut merupakan gambaran proses aplikasi yang akan dibangun. dapat dilihat pada gambar berikut.

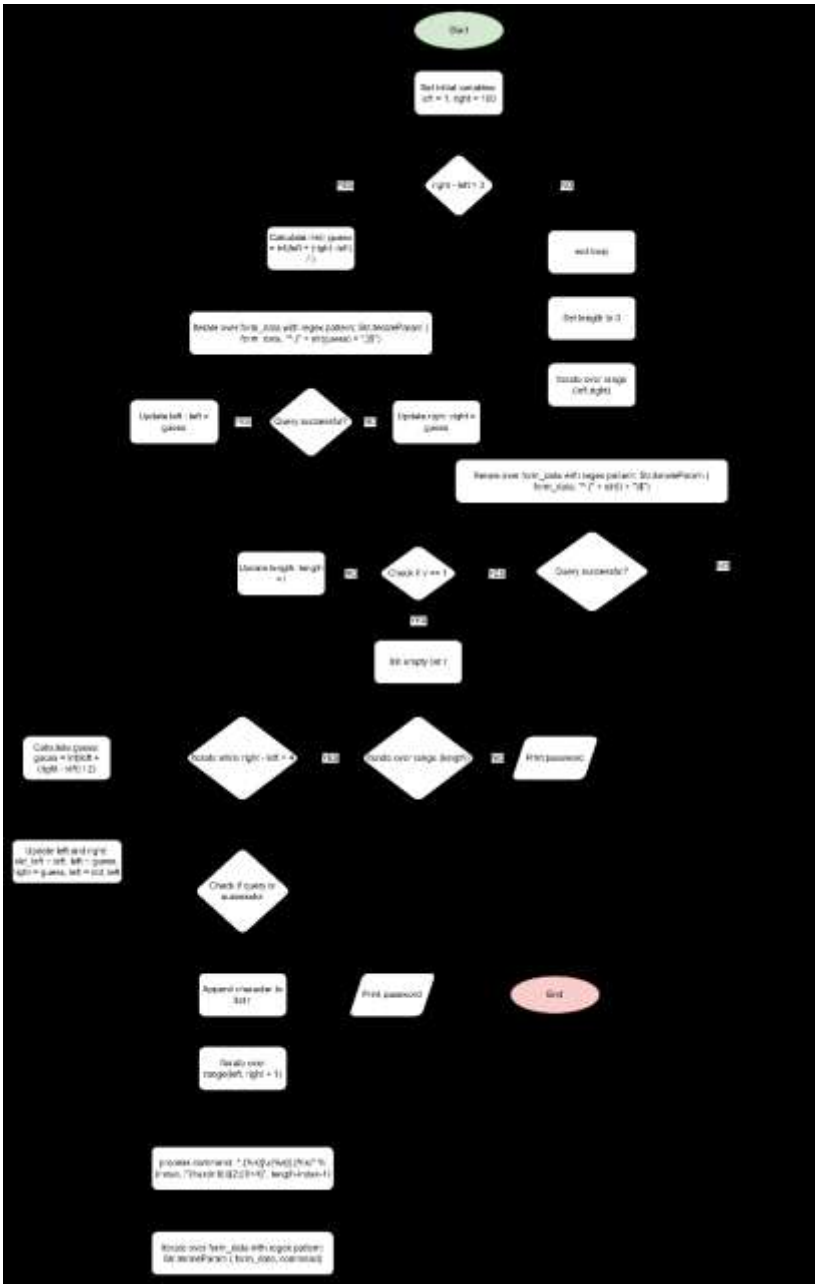


Gambar 3.1 Perancangan Flowchart Sistem

3.2.2.2 Flowchart Algoritma Binary Search

Flowchart ini berisi gambaran proses dari eksploitasi dengan pendekatan Algoritma *Binary Search yang* merupakan upaya optimasi dari Algoritma *Linear Search.* Algoritma mencari setiap huruf dalam data dengan membagi data menjadi dua. Kemudian data tersebut dibandingkan apakah hasilnya di atas atau di bawah sampai mendapatkan semua hasil dari setiap huruf yang disusun menjadi sebuah kata. Bentuk dan Respon memiliki fungsi yang sama dalam algoritma pencarian linier.

Berikut merupakan gambaran proses aplikasi yang akan dibangun. dapat dilihat pada gambar berikut.

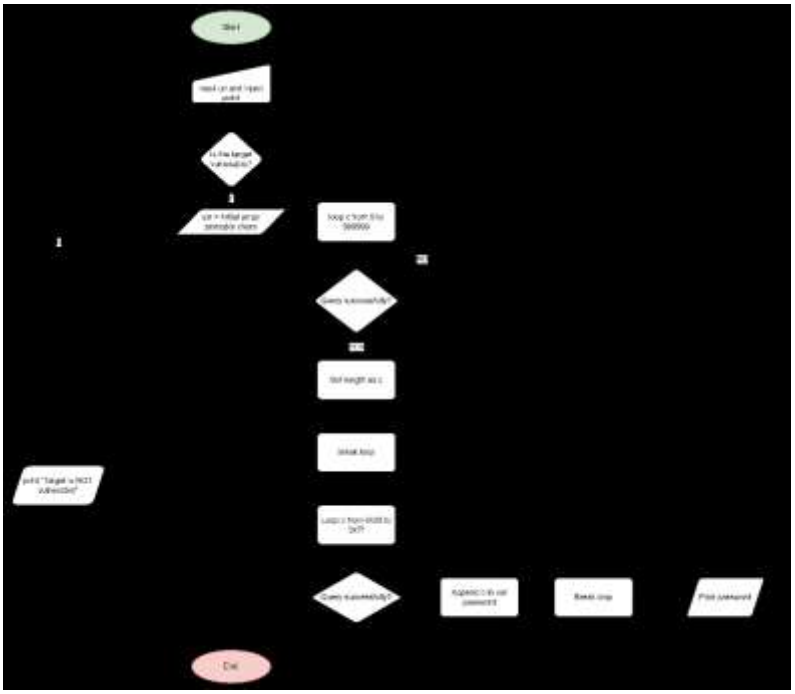


Gambar 3.2 Perancangan Flowchart Algoritma *Binary Search*

3.2.2.3 Flowchart Algoritma Linear Search

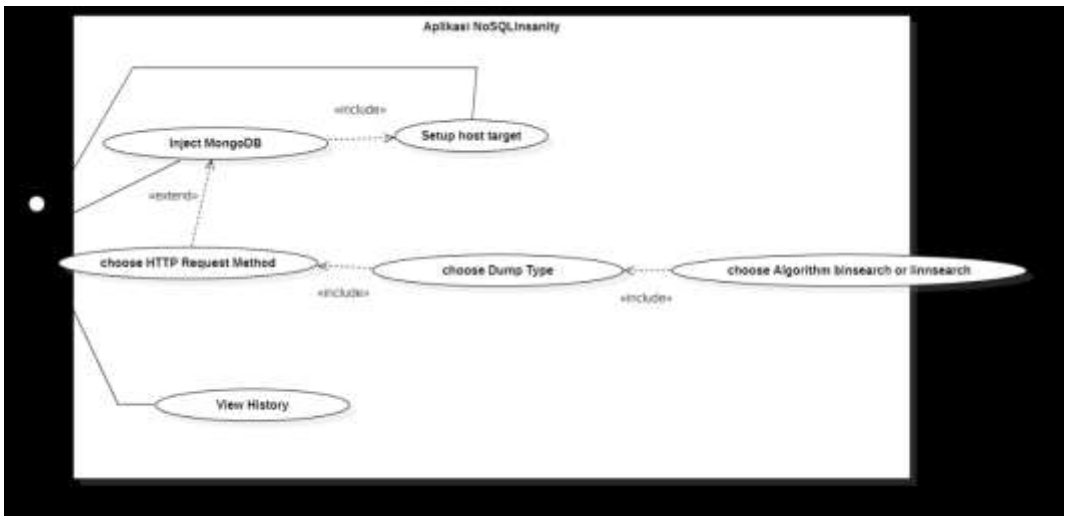
Flowchart ini berisi gambaran proses dari eksploitasi dengan pendekatan Algoritma L*inear Search*. Algoritma mencari setiap huruf yang benar dalam *database* dan kemudian melakukannya berulang kali selama N kali. Perulangan N

kali adalah prediksi panjang char yang disusun menjadi kalimat sehingga setiap kali program berulang dan jawabannya benar. Kemudian menghasilkan huruf yang menunjukkan bahwa surat tersebut adalah data yang dicari. Pencarian menggunakan data hex secara berurutan dari 0x20 hingga 0x7f. *Form* and *Response* berfungsi untuk menghubungkan aplikasi dengan sistem pada *web* sehingga aplikasi dapat memberikan hasil *query* yang dibuat untuk dikirimkan ke *inject point* yang terdapat pada *website* kemudian menembus *database*.



Gambar 3.3 Flowchart Algoritma *Linear Search*

3.2.3 Use Case Diagram



Gambar 3.4 Use Case Diagram Aplikasi *Blind* MongoDB *Injection*

3.2.3.1 Use Case Setup host target

Nama use case *: Setup host target*
Actor : *User*
Tujuan : Menentukan *host* dan *path* sebagai target serang
Deskripsi : Pada fase ini user diharuskan melakukan konfigurasi *host* dan *path* sebagai target

Tabel 3.1 Use Case *Setup host target*

Action	System user
Description	*User* menginputkan host dan path yang vulnerable
Actor	*User*
Goal	Data ter-set kedalam property *class* atau *temp config* untuk digunakan pada fase selanjutnya
Pre-Condition	Pilih nomor 1 menu pada interaktif *shell*
Post-Condition	Masuk ke interaktif *shell* dan kembali ke *main menu*

3.2.3.2 Use Case *Inject* MongoDB

Nama use case : *Inject* MongoDB
Actor : *User*
Tujuan : Melakukan ekstraksi data sebuah kolom yang *vulnerable*
Deskripsi : Aktor memilih algoritma yang digunakan serangan *Blind NoSQL Injection*

Tabel 3.2 Use Case *Inject* MongoDB

Action	System user
Description	*User* memilih menu MongoDB *Injection Attack*
Actor	*User*
Goal	Dapat mengekstrak data pada kolom yang vulnerable
Pre-Condition	Pilih nomor 2 menu pada interaktif *shell*
Post-Condition	Masuk ke interaktif *shell* lanjutan, menu pilih *method* serangan (POST/GET)

3.2.3.3 Use Case Choose HTTP Method

Nama use case : *Choose HTTP Method*
Actor : *User*
Tujuan : Dapat memilih *method HTTP request* yang akan digunakan
Deskripsi : Aktor dapat memilih dengan beberapa *method* HTTP *GET* ataupun *POST*

Tabel 3.3 Use Case DB *Choose HTTP Method*

Action	System user
Description	*User* memilih *HTTP Method*
Actor	*User*
Goal	Tool dapat me-*request* dengan tipe data POST atau GET

Pre-Condition	Pilih Menu MongoDB *Injection Attack* terlebih dahulu
Post-Condition	Masuk ke interaktif *shell* lanjutan, lalu pilih *POST* atau *GET* Method

3.2.3.4 Use Case Choose Dump Type

Nama use case *: Choose Dump Type*
Actor : *User*
Tujuan : Dapat memilih jenis dump
Deskripsi : Fitur untuk *dump* semua *document* tanpa salah satu *value field* diketahui dan *dump* salah satu *document* dengan salah satu *value field* diketahui

Tabel 3.4 Use Case *Choose Dump Type*

Action	System user
Description	*User* memilih *Dump Type*
Actor	*User*
Goal	Tool dapat melakukan *dumping* document sesuai opsi yang dipilih *user*
Pre-Condition	Pilih Menu *HTTP Method* terlebih dahulu
Post-Condition	Masuk ke interaktif *shell* lanjutan, menu pemilihan algoritma

3.2.3.5 Use Case Choose Algorithm

Nama use case *: Choose Algorithm*
Actor : *User*
Tujuan : Dapat memilih opsi algoritma yang akan digunakan sebagai proses eksploitasi
Deskripsi : Aktor dapat memilih Algoritma *Linear Search* atau *Binary Search*.

Tabel 3.5 Use Case *Choose Algorithm*

Action	System user
Description	*User* memilih algoritma yang akan digunakan
Actor	*User*
Goal	Dapat memilih opsi algoritma yang akan digunakan
Pre-Condition	Pilih menu MongoDB *Injection Attack,* pilih *HTTP method*, masukkan *parameter inject point*
Post-Condition	*Tool* melakukan *request* ke *server target* dengan *payload* yang telah di*generate*

3.2.3.6 Use Case View History

Nama use case *: View History*
Actor : *User*
Tujuan : Dapat melihat history pengujian Blind NoSQL Injection
Deskripsi : Aktor dapat melihat history dari pengujian yang pernah dilakukan sebelumnya.

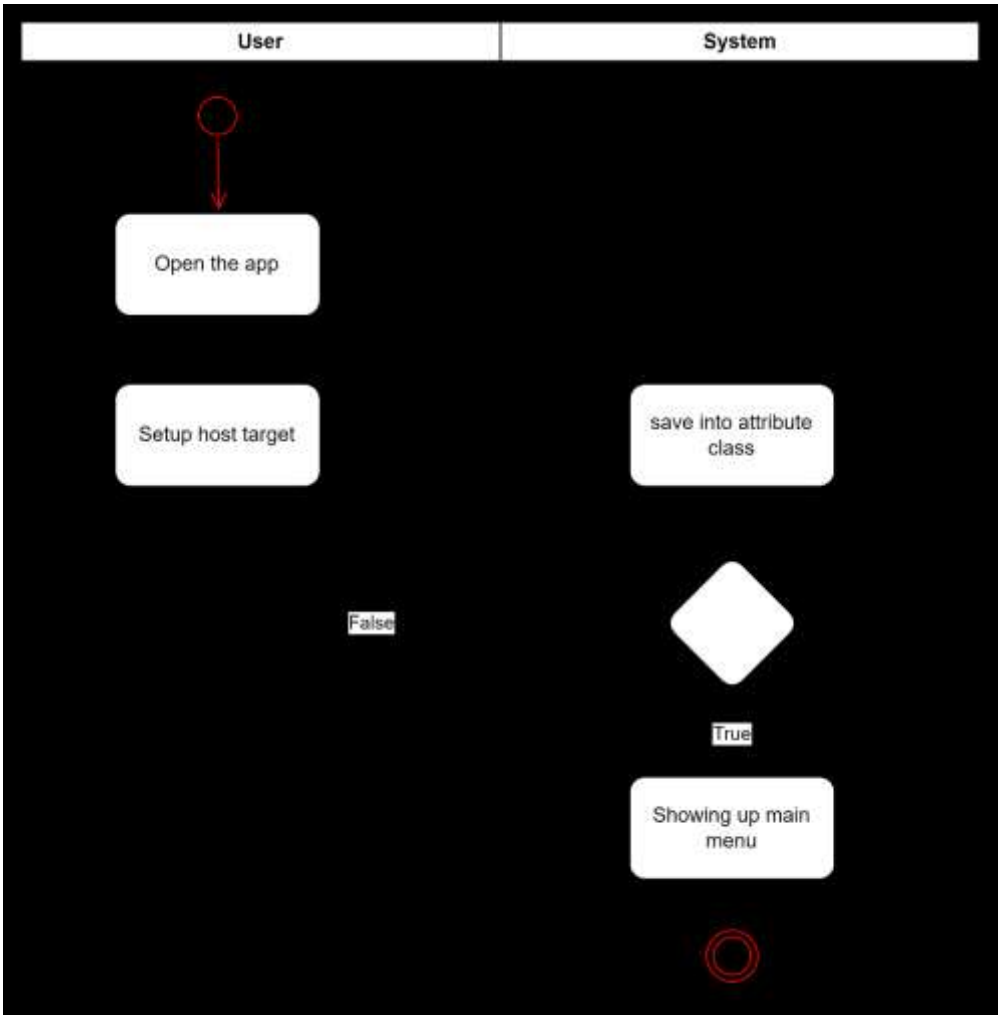
Tabel 3.6 Use Case *View History*

Action	System user
Description	*User* melihat history pengujian
Actor	*User*
Goal	Dapat melihat detail *history* pengujian
Pre-Condition	Pilih menu no. 3 pada interaktif *shell*
Post-Condition	*Tool* menampilkan *detail history* yang pernah dilakukan sebelumnya.

3.2.4 Activity Diagram

3.2.4.1 Activity Diagram *Setup Host Target*

Activity Diagram : *Setup Host Target*
Actor : User
Deskripsi : Aktor membuka aplikasi lalu pilih menu "*Setup host target*" lalu memasukkan data string. Apabila sudah selesai sistem akan meyimpan data secara *temporary*



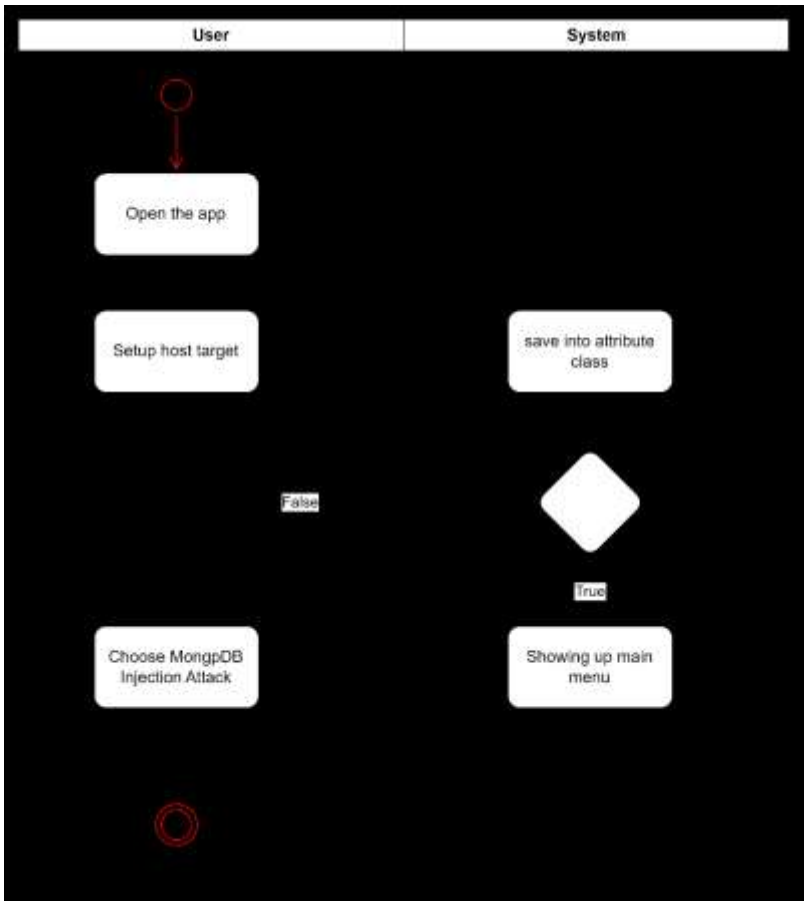
Gambar 3.5 Activity Diagram *Setup Host Target*

3.2.4.2 Activity Diagram *Inject* MongoDB

Activity Diagram : *Inject* MongoDB

Actor : Sistem

Deskripsi : Aktor membuka aplikasi lalu pilih menu "Setup host target" lalu memasukkan data string. Apabila sudah selesai sistem akan meyimpan data secara *temporary* pada *attribute class*, selanjutnya sistem masuk ke menu lanjutan. Selanjutnya, Aktor memilih choose MongoDB *Injection Attack*



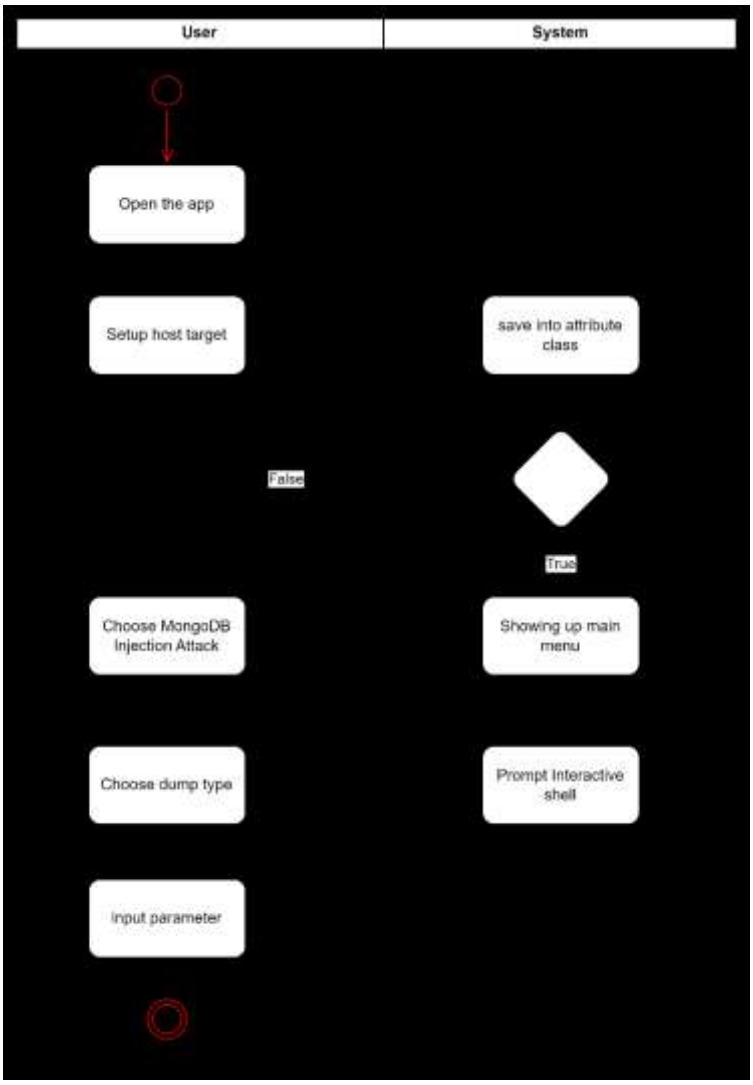
Gambar 3.6 Activity Diagram *Inject* MongoDB

3.2.4.3 Activity Diagram *Choose Dump Type*

Activity Diagram : *Choose Dump Type*

Actor : User

Deskripsi : Aktor membuka aplikasi dan *setup host target* terlebih dahulu lalu pada main menu, aktor memilih *MongoDB Injection Attack*. Kemudian Aktor memilih HTTP Method yang akan digunakan dan sistem merefleksikan ke layar. Selanjutnya pada interaktif *shell* aktor memilih opsi jenis *dump*



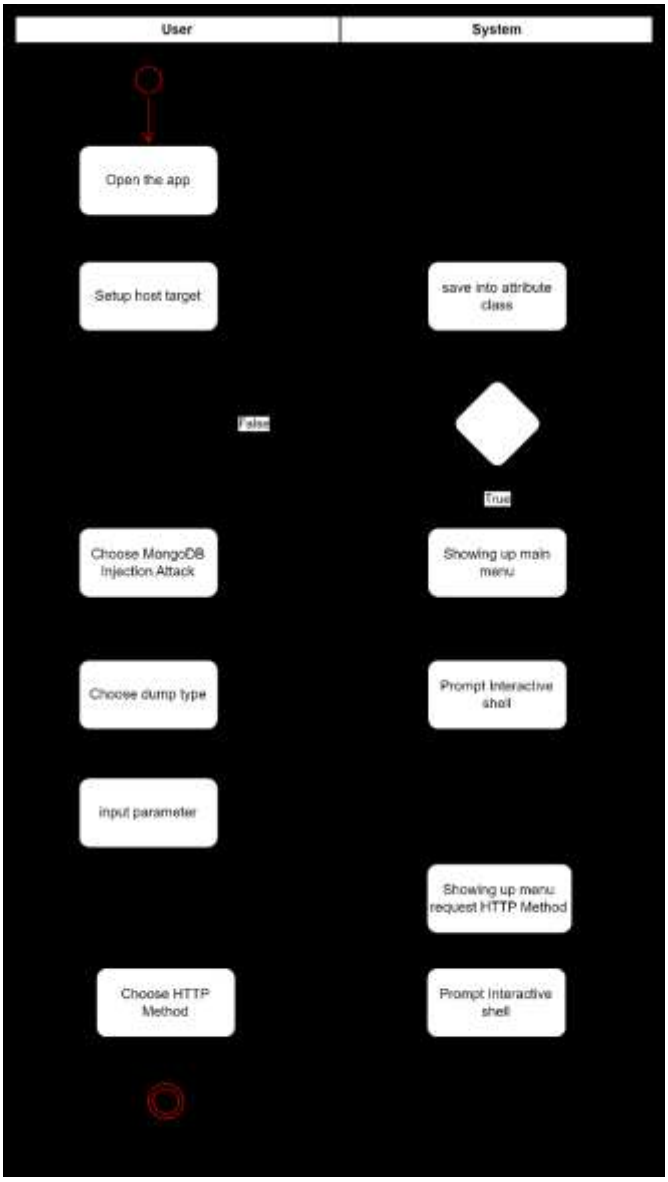
Gambar 3. 7 Activity Diagram *Dump Type*

3.2.4.4 Activity Diagram Choose HTTP Method

Activity Diagram : DB Authentication Attack

Actor : User

Deskripsi : Aktor membuka aplikasi dan setup host target terlebih dahulu lalu aktor memilih *MongoDB Injection Attack*. Selanjutnya Aktor memilih HTTP Method yang akan digunakan dan sistem merefleksikan ke layar.



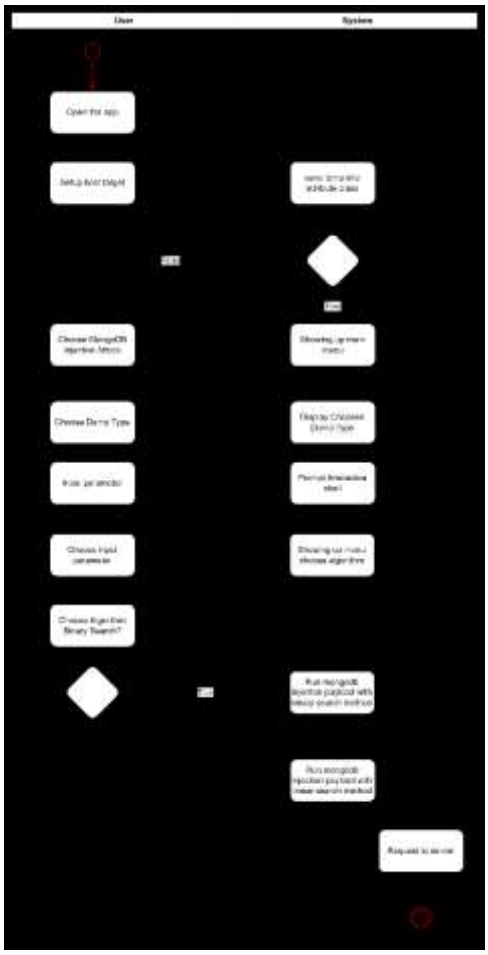
Gambar 3.8 Activity Diagram *Choose HTTP Method*

3.2.4.6 Activity Diagram Choose Algorithm

Activity Diagram : *Choose Algorithm*

Actor : User

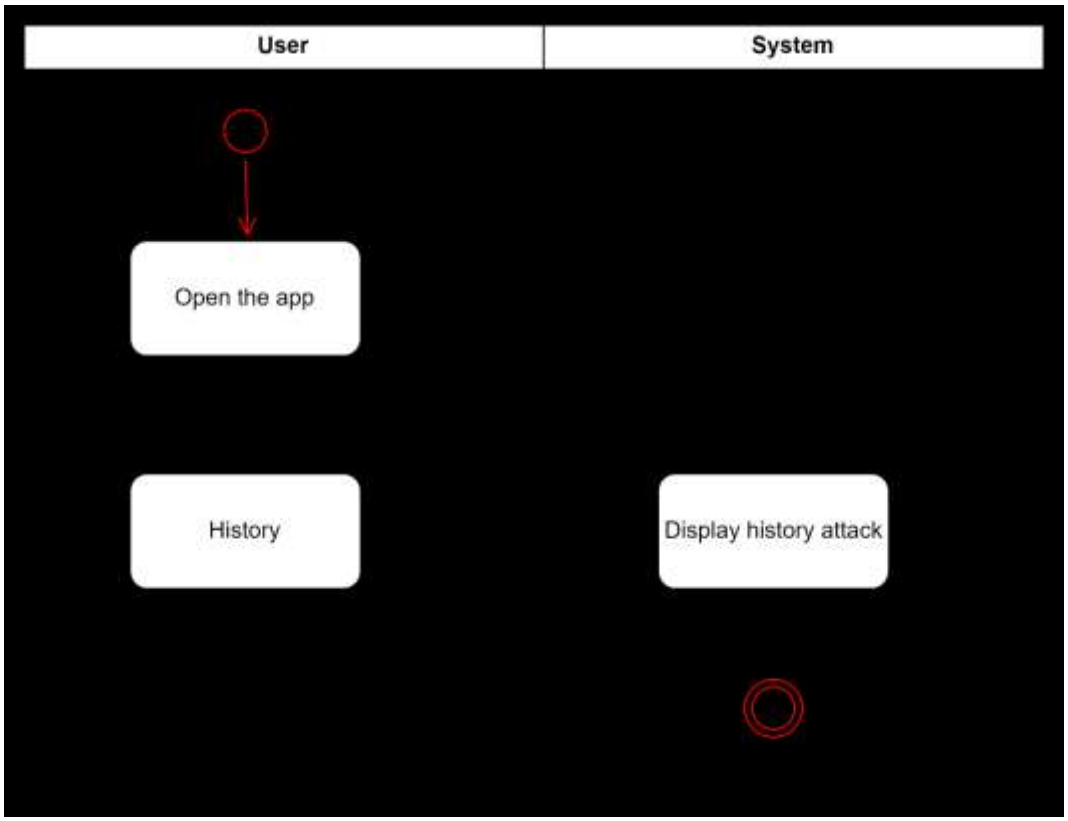
Deskripsi : Aktor membuka aplikasi dan setup host target terlebih dahulu lalu pada main menu, aktor memilih MongoDB Injection Attack. Kemudian Aktor memilih HTTP Method yang akan digunakan dan sistem merefleksikan ke layar. Selanjutnya pada interaktif shell aktor menginputkan parameter yang vulnerable, lalu muncul interaktif shell untuk memilih algoritma yang akan digunakan



Gambar 3.9 Activity Diagram *Choose Algorithm*

3.2.4.6 Activity Diagram History

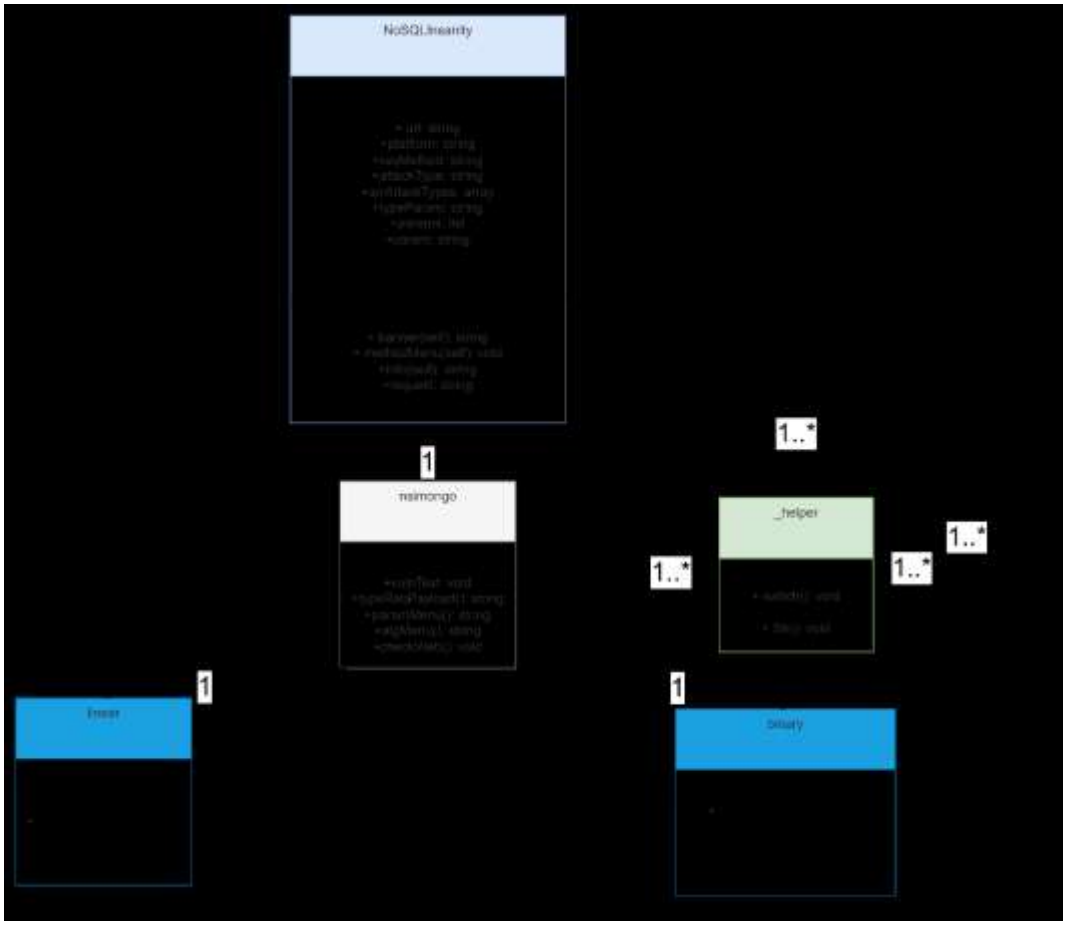
Activity Diagram : *History*
Actor : User
Deskripsi : Aktor membuka aplikasi lalu pada main menu memilih *History* dan sistem akan menampilkan data-data *history* serangan yang pernah dilakukan sebelumnya.



Gambar 3.10 Activity Diagram *History*

3.2.5 Class Diagram

Class diagram pada perancangan aplikasi yang akan dibangun pada gambar 3.11 menampilkan hubungan antara kelas-kelas yang ada.

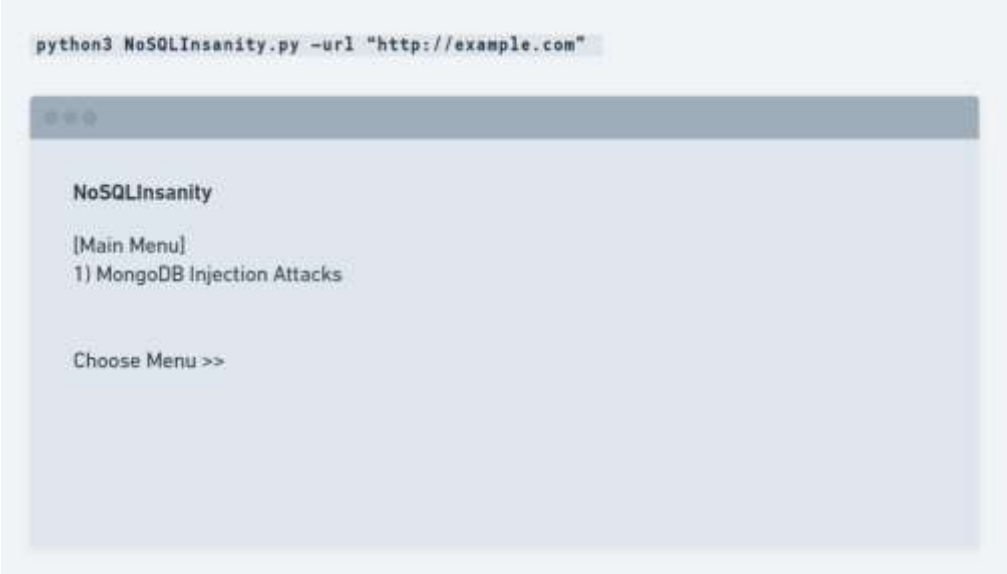


Gambar 3.11 Class Diagram Aplikasi *Blind* NoSQL *Injection*

3.2.6 Perancangan User Interface / *Mock-up* Aplikasi

Pada perancangan *user interface* (UI) memuat beberapa rancangan halaman. User interface ini berguna untuk menggambarkan interaksi antara pengguna dengan sistem yang dibuat. Beberapa rancangan antarmuka yang digunakan dalam sistem ini sebagai berikut.

a. Main Menu



Gambar 3.12 Desain Antarmuka *Main Menu*

Pada halaman ini digunakan untuk menampilkan menu utama sebelum masuk ke menu di interaktif *shell* selanjutnya.

b. Menu Pemilihan HTTP Method

Gambar 3. 13 Desain Antarmuka Menu Pemilihan *HTTP Method*

Desain halaman Pemilihan HTTP Method digunakan untuk user memilih opsi HTTP Method yaitu jenis *request* POST atau GET dengan menginputkan angka.

c. Menu Choose *Dump Type*

Gambar 3. 14 Desain Antarmuka Menu *Choose Dump Type*

Pada halaman ini digunakan untuk menampilkan pilihan jenis *dumping data* yang akan digunakan terdapat 2 submenu yaitu:

1. *Dump data without known value*: digunakan untuk melakukan *dumping* data pada semua *document* tanpa mengetahui value
2. Dump data by known value: digunakan untuk melakukan *dumping* data pada salah satu *document*

d. Menu Choose Algorithm

Gambar 3. 15 Desain Antarmuka Menu *Choose Algorithm*

Pada halaman ini digunakan untuk fasilitas user memilih algoritma apa yang akan digunakan untuk proses injeksi, diantaranya *Linear search* dan *Binary search.*

3.3 Rancangan Pengujian

Pada penelitian ini, metode pengujian yang akan digunakan untuk mengembangkan aplikasi ini adalah *greybox testing*. *Greybox testing* atau bisa disebut tes fungsional ini adalah pengujian yang dilakukan dengan mengamati hasil eksekusi melalui kasus uji dan memeriksa fungsional dari aplikasi yang sedang dikembangkan. Pengujian program aplikasi ini dilakukan oleh pengembang memberi data yang akan diinputkan. Selain itu pengembang mencoba berbagai fitur dan subfungsi pada aplikasi ini. Hal-hal yang menjadi keutamaan dalam pengujian adalah sebagai berikut:

A. Aplikasi dapat melakukan eksploitasi *MongoDB Injection Attack* dan menjalankan interaktif *shell*
B. Aplikasi dapat melakukan eksploitasi dan memberikan *response* data hasil eksfiltrasi dari proses injeksi ke *database*.
C. Kedua algoritma dapat dikomparasi dari segi waktu proses injeksi dengan menampilkan *timestamp*.

Untuk memudahkan penjelasan dibagi dengan beberapa bagian diantaranya.

3.3.1 Pengujian Serangan Blind NoSQL Injection

Pengujian dilakukan terhadap lab yang memiliki fitur login admin dan menampilkan data post. Pengujian dilakukan dengan menginputkan *malicious JS code* pada sebuah parameter untuk mengetahui apakah host target *vulnerable* terhadap NoSQL Injection sebelum dilanjutkan proses injeksi dengan penerapan algoritma. Berikut sintaksis yang akan digunakan untuk pengujian *vulnerability*.

Tabel 3.7 Sintaksis Pengujian *Vulnerablility*

Sintaks	Expected Result	Deskripsi

Sintaks	Expected Result	Deskripsi
[$ne]	Halaman menampilkan *dashboard* admin	Sintaks disamping diartikan sebagai "not equal / tidak sama dengan" jika website memberikan *return* post kembali normal artinya *statement* NoSQL tersebut *True.* Sebaliknya, maka *statement* NoSQL bernilai *False.*
[$gt]	Halaman menampilkan *dashboard* admin	Sintaks disamping diartikan sebagai "*greater than* / lebih besar dari" jika website memberikan *return* post kembali normal artinya *statement* NoSQL tersebut *True.* Sebaliknya, maka *statement* NoSQL bernilai *False.*

Sintaks	Expected Result	Deskripsi
^.{x}[\x{**hexvalue**}-\x{ **hexvalue** }].{y}	Masuk halaman dashboard	Sintaksis disamping merupakan ekspresi regex digunakan untuk mengecek rentang/*range* dari *substring* suatu kata
^.{x,}$	Masuk halaman dashboard	Sintaksis disamping merupakan ekspresi regex digunakan untuk mengecek apakah panjang kata lebih dari X

Sintaks	Expected Result	Deskripsi
^.{x}$	Masuk halaman dashboard	Sintaksis disamping merupakan ekspresi regex digunakan untuk mengecek apakah panjang kata sama dengan X
^(?!.*abcdefghijklm) ^%s([\x{**hexvalue**}- \x{**hexvalue**}])	Masuk halaman dashboard	Sintaksis disamping merupakan ekspresi regex digunakan untuk mengecek rentang/*range* dari *substring* suatu kata dengan kondisi eksklusi dari kata yang telah ditemukan sebelumnya.

3.3.2 Pengujian Algoritma

Pengujian dilakukan terhadap password dengan panjang 11 karakter. Pengujian dilakukan berdasarkan waktu pencarian untuk setiap karakter. Agar pencarian seimbang dan adil, masing-masing algoritma mencari target dengan skenario yang sama yaitu karakter dari susunan kata. Pengujian dilakukan antara

Algoritma *Linear Search* dan *Binary Search* untuk dapat mengetahui kekuatan dari masing-masing algoritma. Pada saat pengujian akan dicetak *timestamp* dari setiap mendapatkan masing-masing karakter.