Aplikasi Teori Kombinatorial dan Kompleksitas Algoritma dalam Perbandingan Teknik Injeksi SQL Berbasis Boolean

Steve Bezalel Iman Gustaman - 13518018
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13518018@std.stei.itb.ac.id

***Abstrak*—Injeksi SQL adalah suatu penyerangan terhadap keamanan basis data dalam aplikasi berbasis *web*. Pada injeksi SQL berbasis boolean, terdapat beberapa teknik yang dapat digunakan yang membutuhkan adanya penebakan. Dalam makalah ini akan dibahas mengenai perbandingan beberapa teknik injeksi SQL berbasis boolean dilihat dari jumlah penebakan maksimum yang dibutuhkan melalui pendekatan teori kombinatorial, serta kompleksitas algoritma penebakan tersebut.**

***Kata kunci*—injeksi SQL, keamanan basis data, teori kombinatorial, kompleksitas algoritma.**

I. PENDAHULUAN

Dewasa ini, pesatnya perkembangan teknologi adalah suatu hal yang sudah tidak dapat dipungkiri lagi. Dampak nyata dari kecepatan berkembangan teknologi ini adalah pengguna internet yang banyak dan semakin meningkat. Pada internet sendiri pastinya tersimpan data yang jumlahnya tak terhitung. Penyimpanan data ini dilakukan dalam suatu basis data. Suatu aplikasi dalam internet dapat mengakses dan mengubah data dalam basis data menggunakan sistem yang disebut dengan sistem manajemen basis data. Cara kerja sistem ini adalah dengan komunikasi menggunakan kueri (*query*) yang diibaratkan sebagai permintaan terhadap basis data. Salah satu bahasa yang kerap digunakan dalam pengaturan kueri dalam sistem manajemen basis data adalah SQL.

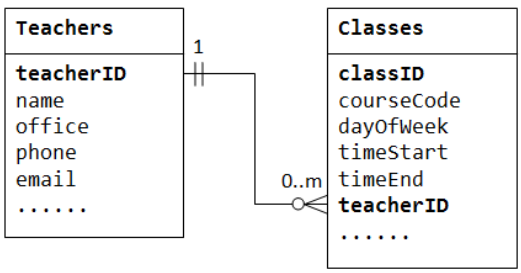
Seperti yang sudah dijelaskan, dalam internet tersimpan data yang sangat banyak. Dari sekian banyaknya data yang tersimpan, beberapa di antaranya dapat berupa data yang sensitif bagi para penggunanya, semisal sandi untuk suatu akun. Karenanya, keamanan penyimpanan data menjadi suatu hal yang penting. Salah satu penyerangan yang dapat dilakukan terhadap basis data yang keamanannya tidak teruji adalah penyerangan injeksi SQL. Penyerangan ini tentunya dapat dimanfaatkan untuk kejahatan, namun di sisi lain, penyerangan ini dapat dimanfaatkan oleh para penguji keamanan sebagai alat untuk menguji keamanan aplikasi. Dengan serangan ini, pelaku serangan bisa mendapatkan data dalam basis data yang bersangkutan. Jenis dari penyerangan ini juga ada beberapa macam, yang akan dibahas pada makalah ini adalah penyerangan berbasis boolean.

Pada macam penyerangan ini dibutuhkan penebakan. Semakin banyak penebakan yang dilakukan, maka serangan yang dilakukan menjadi semakin lambat dan tidak efektif. Dalam makalah ini, akan dibandingkan beberapa teknik injeksi SQL berbasis boolean dilihat dari jumlah penebakan maksimum yang harus dibuat serta kompleksitas dari algoritma teknik penebakan untuk mengetahui teknik yang paling efektif. Perhitungan jumlah tebakan maksimum akan dilakukan dengan pendekatan teori kombinatorial dan algoritma teknik penebakan akan dianalisis dengan pendekatan kompleksitas algoritma.

II. TEORI DASAR

A. Basis Data dan SQL

Basis data adalah kumpulan informasi yang terstruktur, disimpan dalam suatu sistem komputer. Suatu basis data biasanya diatur dalam suatu sistem yang dinamakan sistem manajemen basis data atau *database management system (DBMS)*. Basis data, DBMS, serta aplikasi-aplikasi lain yang bersangkutan disebut sebagai satu sistem basis data. Data yang disimpan ini umumnya dimodelkan dalam bentuk baris dan kolom dalam beberapa tabel. Kebanyakan dari sistem basis data dalam aplikasi berbasis *web* menggunakan basis data relasional. Dalam basis data relasional, tiap baris dalam tabel memiliki kunci masing-masing yang memudahkan hubungan antar data dalam basis data tersebut.



Gambar 1. Contoh Ilustrasi Basis Data Relasional
Sumber: https://www.ntu.edu.sg/home/ehchua/programming/sql/Relational_Database_Design.html

SQL atau *Stuctured Query Language* adalah bahasa yang dipakai untuk membaca, memanipulasi, mendefinisikan, serta menghapus data yang ada dalam basis data. Dalam SQL, digunakan suatu perintah untuk berkomunikasi dengan basis data itu sendiri yang dinamakan kueri (*query*). SQL sendiri banyak digunakan dalam berbagai DBMS seperti MySQL, PostgreSQL, SQLite, dan lain-lain. Contoh yang akan digunakan adalah dalam MySQL.

Kueri dapat digunakan untuk membaca (dan mencari) data yang terdapat dalam tabel pada suatu basis data. Salah satu bentuk kueri yang umum digunakan dapat dilihat pada gambar 2 berikut. Salah satu aplikasi dari kueri ini adalah untuk fitur masuk (*log in*) dalam suatu laman *web* dengan memanfaatkan kueri untuk pencarian nama akun dan sandi yang tepat. Contoh aplikasi bentuk ini dapat dilihat pada gambar 3 dengan warna merah melambangkan masukan pengguna berupa nama akun dan sandi pengguna.

```
SELECT seleksi FROM tabel WHERE kondisi;
```

Gambar 2. Contoh bentuk kueri MySQL

```
SELECT * FROM user WHERE username = 'abc'
AND password = 'def';
```

Gambar 3. Contoh aplikasi kueri MySQL

B. Injeksi SQL Berbasis Boolean

Injeksi SQL (*SQL Injection*) adalah suatu serangan terhadap basis data dalam suatu aplikasi berbasis *web*. Serangan ini dapat terjadi karena adanya masukan pengguna yang tidak ditangani dengan baik, dalam hal ini, masukan dari pengguna langsung digabungkan dengan kueri SQL dari aplikasi ke basis data. Pada gambar 4, dapat dilihat contoh injeksi SQL dalam contoh fitur *log in* suatu aplikasi. Warna merah dalam gambar tersebut melambangkan masukan pengguna. Perlu diketahui bahwa kode " -- " dalam kueri MySQL berarti komentar, dengan kata lain, apapun yang ada setelah kode ini tidak akan dianggap sebagai kueri, namun hanya sebagai komentar. Perhatikan bahwa kueri ini akan selalu mengembalikan data yang sahih karena kondisi pencarian akan selalu terpenuhi. Ini akan menyebabkan pengguna dapat *log in* dengan akun yang bukan miliknya.

```
SELECT * FROM user WHERE username =
'hello' OR 1=1 -- ' AND password = 'def';
```

Gambar 4. Contoh injeksi SQL

Injeksi SQL dalam kueri *log in* ini termasuk injeksi SQL buta (*blind SQL injection*). Ini karena data yang dikembalikan kueri (nama dan sandi) tidak ditampilkan, pengguna hanya mengetahui apakah *log in* berhasil atau tidak. Karena itu, data dalam basis data tidak secara langsung dapat dibaca. Dalam injeksi SQL buta, salah satu bentuk injeksi yang dapat digunakan adalah injeksi berbasis boolean. Dengan bentuk injeksi ini, pengguna hanya perlu mengetahui apakah ada nilai yang dikembalikan oleh kueri atau tidak. Karena nilai ya atau tidak ini, bentuk injeksi ini disebut injeksi berbasis boolean. Dalam injeksi ini, diperlukan adanya penebakan, ada beberapa teknik penebakan yang dapat digunakan.

Salah satu data yang sensitif dalam basis data adalah nama basis data yang digunakan itu sendiri, karena dengan mengetahui nama basis data, basis data tersebut dapat dieksploitasi lebih lanjut. Pada gambar 5 dicontohkan penebakan nama basis data menggunakan bentuk injeksi berbasis boolean. Perhatikan bahwa karena terdapat ekspresi *AND FALSE* pada bagian pengcekan *username* serta kode komentar pada bagian pengecekan *password*, kueri ini akan mengembalikan nilai jika dan hanya jika basis data yang digunakan bernama 'login'.

```
SELECT * FROM user WHERE username =
'hello' AND FALSE OR database() = 'login'
-- ' AND password = 'def';
```

Gambar 5. Contoh injeksi SQL berbasis boolean

C. Teori Kombinatorial

Kombinatorial adalah salah satu cabang matematika yang mempelajari cara menghitung jumlah penyusunan objek-objek tanpa harus mengenumerasi semua kemungkinan susunannya. [5] Dalam teori kombinatorial, dikenal dua buah kaidah dasar untuk perhitungan yaitu kaidah perkalian (*rule of product*) dan kaidah penjumlahan (*rule of sum*).

Kaidah perkalian digunakan apabila terdapat dua buah percobaan dengan jumlah hasil masing-masing, dan akan dihitung hasil percobaan pertama dan percobaan kedua. Misalnya percobaan pertama mungkin menghasilkan sejumlah p hasil dan percobaan kedua mungkin menghasilkan sejumlah q hasil. Maka, percobaan pertama dan kedua, mungkin menghasilkan sejumlah $p \times q$ hasil. [5]

Salah satu contoh penggunaan kaidah perkalian adalah dalam penentuan banyaknya sandi lewat yang sahih. Misalkan karakter sandi yang valid adalah huruf 'a' sampai 'z' (26 huruf), dan sandi ini terdiri dari 5 huruf. Perhatikan bahwa terdapat 26 cara untuk memilih setiap huruf dalam setiap posisi (1 sampai 5). Maka, untuk banyaknya sandi 5 huruf yang sahih adalah Banyaknya huruf pada posisi pertama yang valid (26) dan banyaknya huruf pada posisi kedua yang valid (26), seterusnya sampai posisi kelima. Dengan kaidah perkalian didapatkan sejumlah $26 \times 26 \times 26 \times 26 \times 26 = 26^5 = 11881376$ kemungkinan kombinasi sandi.

Kaidah dasar lain dalam teori kombinatorial adalah kaidah penjumlahan. Kaidah ini digunakan apabila terdapat dua buah percobaan dengan jumlah hasil masing-masing, dan yang akan dihitung adalah hasil percobaan pertama atau percobaan kedua. Misalkan percobaan pertama mungkin menghasilkan sejumlah p hasil berbeda dan percobaan kedua mungkin menghasilkan q hasil berbeda. Maka, percobaan pertama atau kedua, mungkin menghasilkan $p + q$ hasil. [5]

Contoh dari penggunaan kaidah penjumlahan adalah dalam pencarian banyaknya karakter yang mungkin dalam suatu sandi lewat. Misalnya dalam suatu sandi lewat, karakter yang diperbolehkan adalah huruf kecil ('a' sampai 'z'), huruf kapital ('A' sampai 'Z'), dan angka ('0' sampai '9'). Maka, jumlah karakter yang valid ada sejumlah karakter huruf kecil yang valid (26), atau karakter huruf kapital yang valid (26), atau angka yang valid (10). Dengan kaidah penjumlahan, didapatkan sejumlah $26 + 26 + 10$, atau 62 karakter yang valid.

Kedua kaidah ini sangat mungkin untuk digunakan secara bersamaan dalam perhitungan. Misalnya dalam perhitungan banyaknya sandi lewat 5 huruf yang valid diberikan karakter yang valid adalah semua karakter huruf kecil, huruf kapital, dan angka. Sama dengan contoh sebelumnya, kaidah penjumlahan digunakan untuk mencari banyaknya karakter total yang valid untuk satu huruf, lalu kaidah perkalian digunakan untuk menentukan total semua kemungkinan sandi yang valid. Dari kedua kaidah, didapatkan jumlah sandi yang mungkin sebanyak $62^5 = 916132832$ kombinasi.

Jumlah karakter yang valid	26	26	10
Kategori	Huruf kecil	Huruf kapital	Angka

Jumlah huruf yang mungkin	62	62	62	62	62
Posisi	1	2	3	4	5

Gambar 6. Ilustrasi contoh kaidah perkalian dan penjumlahan

Dalam teori kombinatorial, dikenal juga istilah permutasi dan kombinasi. Permutasi dan kombinasi adalah perluasan dari kaidah-kaidah dasar yang telah dibahas sebelumnya. Permutasi adalah jumlah urutan berbeda dari pengaturan objek-objek. Permutasi dapat dipahami juga sebagai bentuk khusus aplikasi kaidah penjumlahan dalam pengurutan. Dalam pengurutan n objek, posisi pertama dapat diisi dengan n objek, posisi kedua dapat diisi dengan $(n-1)$ objek (karena 1 objek sudah menmpati posisi pertama) demikian seterusnya sampai posisi yang terakhir. Ilustrasi dapat dilihat pada gambar 7. Dalam notasinya, permutasi r dari n elemen dapat didefinisikan sebagai kemungkinan urutan r elemen dipilih dari n elemen tanpa elemen yang berulang. [5]

$$P(n,r) = n(n-1)(n-2)\dots(n-(r-1)) = \frac{n!}{(n-r)!}$$

Jumlah objek yang mungkin	n	n-1	n-2	n-3	…
Posisi	1	2	3	4	…

Gambar 7. Ilustrasi permutasi sebagai bentuk khusus kaidah perkalian dalam pengurutan

Kombinasi dapat didefinisikan sebagai banyaknya pengaturan objek-objek dengan urutan kemunculan tidak diperhitungkan. Kombinasi dapat juga dipahami sebagai permutasi dengan elemen-elemen yang mengandung objek-objek yang sama (meskipun urutan kemunculannya berbeda) dianggap sama. Dalam permutasi r dari n elemen, terdapat $r!$ cara untuk membentuk elemen dengan objek-objek yang sama. Agar elemen-elemen tersebut dianggap sama, maka hasil permutasi perlu dibagi dengan $r!$. Dalam notasinya, kombinasi r dari n elemen adalah jumlah pemilihan tidak terurut r buah elemen yang diambil dari n buah elemen. [5]

$$C(n,r) = \frac{n(n-1)(n-2)\dots(n-(r-1))}{r!} = \frac{n!}{r!\,(n-r)!}$$

D. Kompleksitas Algoritma

Kemangkusan suatu algoritma dapat dihitung dari waktu yang dibutuhkan untuk mengeksekusi algoritma tersebut. Namun, dengan cara ini, algoritma yang sama pada arsitektur mesin yang berbeda sangat mungkin menghasilkan waktu yang berbeda. Oleh karena itu, dibutuhkan suatu model perhitungan dan pengukuran waktu serta ruang yang independen terhadap mesin. Inilah yang disebut dengan kompleksitas algoritma.

Dalam kompleksitas algoritma, terdapat dua macam besaran yang dapat dihitung, yaitu kompleksitas waktu dan ruang. Dalam makalah ini hanya akan dibahas mengenai kompleksitas waktu. Kompleksitas waktu, biasa disimbolkan $T(n)$, adalah jumlah tahapan komputasi yang dibutuhkan untuk menjalankan suatu algoritma sebagai fungsi dari masukannya, yaitu n. Dengan menggunakan besaran ini, pemrogram dapat menghitung dan menentukan laju yang diperlukan dengan meningkatnya masukan (n) dilihat dari ukurannya. [6]

Dalam praktiknya, kompleksitas waktu suatu algoritma dihitung dari jumlah dari operasi khas suatu algoritma yang dilakukan. Sebagai contoh, yang menjadi operasi khas algoritma pencarian adalah operasi perbandingan (*comparison*), dan yang menjadi operasi khas dari algoritma penjumlahan matriks adalah penjumlahan elemen matriks yang bersangkutan. Jika dalam suatu algoritma pencarian jumlah operasi perbandingan yang dilakukan adalah n kali. Maka, dapat disebutkan bahwa $T(n)$ untuk algoritma tersebut bernilai n.

Pada analisis kompleksitas algoritma, terdapat istilah kompleksitas waktu asimptotik atau yang lebih dikenal dengan notasi "O-Besar" (*Big-O notation*). Secara definisi, $T(n) = O(f(n))$ jika terdapat konstanta C dan n_0 sedemikian sehingga $T(n) \leq C(f(n))$ untuk $n \geq n_0$, atau dengan kata lain, $f(n)$ adalah batas atas dari $T(n)$ untuk suatu n yang besar. [6] Sebagai contoh, jika $T(n)$ terdefinisi sebagai,

$$T(n) = 2n^2 + 6n + 1.$$

Maka, kompleksitas waktunya dalam notasi "O-Besar" adalah $O(n^2)$. Karena untuk $C = 9$ dan $n_0 = 1$,

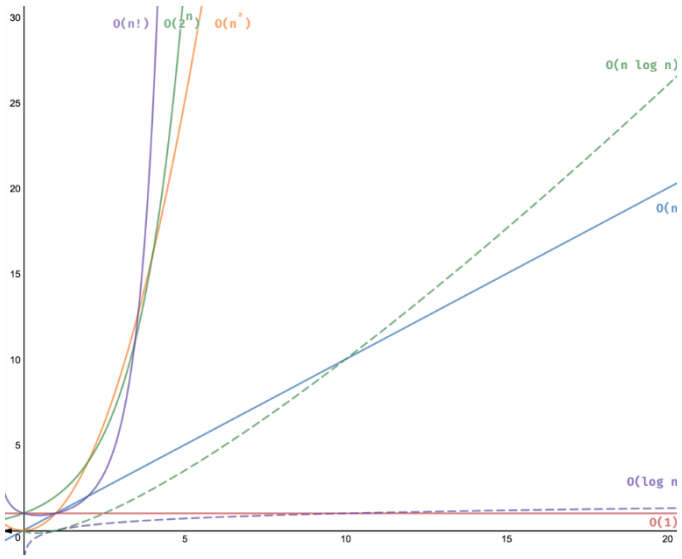
$$2n^2 + 6n + 1 \leq 2n^2 + 6n^2 + n^2 = 9n^2$$

Pada tabel 1 dapat dilihat beberapa pengelompokan algoritma dari notasi "O-Besar"-nya.

Kelompok Algoritma	**Nama**
$O(1)$	Konstan
$O(\log n)$	Logaritmik
$O(n)$	Lanjar
$O(n \log n)$	n log n
$O(n^2)$	Kuadratik
$O(n^3)$	Kubik
$O(2^n)$	Eksponensial
$O(n!)$	Faktorial

Tabel 1. Pengelompokan algoritma berdasarkan notasi "O-Besar"

Sedangkan spektrum dari kompleksitas waktu algoritma ini, $O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n) < O(n!)$. Spektrum ini dapat dilihat lebih jelas pada gambar 8.



Gambar 8. Spektrum kompleksitas waktu algoritma sesuai pengelompokkan notasi "O-Besar"
Sumber: https://adrianmejia.com/most-popular-algorithms-time-complexity-every-programmer-should-know-free-online-tutorial-course/

III. Teknik-teknik Injeksi SQL Berbasis Boolean

A. Teknik Brute-Force

Teknik injeksi SQL dengan *brute-force* adalah teknik di mana penguji menyerang basis data dengan cara menebak seluruh kemungkinan kunci. Dalam makalah ini dicontohkan kunci yang ditebak adalah nama basis data karena alasan yang telah diungkapkan pada bab sebelumnya. Contoh dari serangan menggunakan teknik ini ada pada gambar 5, pada contoh tersebut, tebakan nama basis data yang dibuat adalah 'login'. Kueri akan mengembalikan nilai jika nama basis data sesuai.

B. Teknik Substring

Teknik injeksi SQL dengan *brute-force* tentunya akan memakan waktu yang sangat lama karena kombinasi kunci yang sangat banyak. Teknik lain yang lebih cepat dari teknik tersebut yang dapat dipakai adalah teknik *substring*. Teknik ini memanfaatkan salah satu fungsi dari SQL (secara khusus MySQL), yaitu fugsi *SUBSTRING*. Dengan teknik ini, kunci akan ditebak secara satu karakter per satu karakter. Tentunya dengan penebakan satu per satu ini, jumlah tebakan akan berkurang.

Fungsi *SUBSTRING* digunakan untuk mengambil potongan dari suatu *string*. Bentuk fungsi ini dapat dilihat pada gambar 9 di mana string adalah suatu nama, posisi adalah posisi awal pengambilan potongan, dan panjang adalah panjang karakter potongan.

```
SUBSTRING(string, posisi, panjang);
```

Gambar 9. Contoh bentuk fungsi *SUBSTRING*

```
SUBSTRING('abcde', 3, 1);
```

Gambar 10. Contoh penggunaan fungsi *SUBSTRING*

Contoh penggunaan fungsi ini dapat dilihat pada potongan kode pada gambar 10. Pada contoh tersebut, *string* adalah 'abcde', posisi adalah 3, serta panjang potongan adalah 1. Artinya, *string* 'abcde' dipotong mulai dari posisi 3 sebanyak 1 huruf. Ekspresi tersebut akan mengembalikan nilai 'c'.

```
SELECT * FROM user WHERE username = 'hi'
AND FALSE OR SUBSTRING(database(),2,1) =
'o' -- ' AND password = 'def';
```

Gambar 11. Contoh injeksi SQL berbasis boolean menggunakan teknik *substring*

Dalam injeksi SQL, fungsi *SUBSTRING* dapat digunakan seperti contoh pada gambar 11 di atas. Dapat dilihat bahwa karena ada ekspresi *AND FALSE*, bagian pengecekan *username* tidak akan berpengaruh. Demikian juga halnya, pengecekan *password* tidak akan berpengaruh karena adanya simbol komentar. Nilai *True* hanya akan dikembalikan apabila ekspresi yang berada bagian tengah menghasilkan nilai *True*. Bagian *SUBSTRING(database(), 2, 1)*, digunakan untuk mengambil karakter kedua dari nama basis data, potongan *string* ini akan ditebak dengan cara perbandingan dengan huruf 'o'. Dapat disimpulkan, pada contoh di atas, keseluruhan kode akan menghasilkan nilai *True*, jika dan hanya jika huruf kedua dari nama basis data adalah huruf 'o'. Dengan penebakan karakter dan iterasi posisi karakter, nama basis data bisa didapatkan.

C. Teknik Substring dengan Bit Shifting

Teknik *substring* dalam injeksi SQL berbasis boolean dapat diperluas lebih lanjut sehingga penebakan yang dibutuhkan menjadi berkurang. Fungsi dari SQL (secara khusus MySQL) yang digunakan dalam teknik ini, selain *SUBSTRING* adalah fungsi *ASCII*. Bentuk fungsi ini dapat dilihat pada gambar 12. Fungsi ini digunakan untuk mengembalikan nilai ASCII dari suatu karakter. Contoh pada gambar 13 akan mengembalikan nilai 65, yaitu nilai ASCII dari karakter 'A'. Perlu diperhatikan bahwa nilai ASCII yang valid berupa bilangan bulat 8-bit.

```
ASCII(karakter);
```

Gambar 12. Contoh bentuk fungsi *ASCII*

```
ASCII('A');
```

Gambar 13. Contoh penggunaan fungsi *ASCII*

Teknik ini juga memanfaatkan operator aritmatika *bit shift* secara khusus operator *shift right* yang ekuivalen dengan pembagian bilangan bulat oleh 2^n. Selain itu, digunakan juga operator *and*. Kombinasi dari kedua operator aritmatika dalam bit ini, dapat menjadi alat untuk mengambil suatu bit dari suatu angka. Sebagai contoh, ekspresi $(65 \gg 2) \,\&\, 1$ dapat digunakan untuk mengambil bit ketiga dari kanan dari angka 65. Nilai yang diambil inilah yang kemudian akan dibandingkan dengan nilai 1 atau 0 (kemungkinan bit). Perbandingan ini berhasil dan akan mengembalikan nilai *True* apabila bit pada posisi yang bersangkutan bernilai sama dengan tebakan.

Jika pada teknik sebelumnya kunci ditebak satu karakter per satu karakter, dalam teknik ini, kunci ditebak satu bit per satu bit dengan satu karakter sejumlah 8 bit. Ini dapat mengurangi lebih lanjut jumlah penebakan yang dibutuhkan.

```
SELECT * FROM user WHERE username = 'hi'
AND FALSE OR (((ASCII(SUBSTRING(
database(), 1, 1)) >> 2) & 1)=1) --' AND
password = 'def';
```

Gambar 14. Contoh injeksi SQL berbasis boolean menggunakan teknik *substring* dengan *bit shifting*

Sama dengan contoh-contoh sebelumnya, dalam contoh pada gambar 14 ini, ekspresi *AND FALSE* serta simbol komentar digunakan agar nilai keseluruhan hanya bergantung pada ekspresi injeksinya saja. Dalam hal ini fungsi *SUBSTRING(database(), 1, 1)* mengembalikan karakter pertama dari nama basis data, kemudian fungsi *ASCII* mengembalikan nilai ASCII dari karakter tersebut. Operator *bit shift* dan *and* akan mengambil nilai bit ketiga dari kanan nilai yang dihasilkan. Dalam contoh ini nilai bit yang diambil ditebak dengan nilai 1, maka keseluruhan ekspresi akan mengembalikan nilai boolean *True* jika dan hanya jika bit ketiga dari kanan nilai ASCII dari karakter pertama nama basis data adalah 1. Dengan mengiterasi bit dari nilai ASCII yang diambil serta posisi karakter, hasil tiap percobaan dapat dicatat dan hasilnya (dalam bit) dapat dikembalikan menjadi karakter, maka nama basis data bisa didapatkan.

IV. Perbandingan Teknik Injeksi SQL dengan Pendekatan Teori Kombinatorial

Dalam aplikasinya, teknik-teknik injeksi SQL berbasis boolean membutuhkan adanya penebakan. Jumlah penebakan maksimum yang harus dibuat ini tentunya sangat bergantung pada teknik yang digunakan. Semakin banyak jumlah tebakan yang harus dibuat, maka semakin lambat proses injeksi berjalan. Jumlah ini dapat dihitung dengan bantuan teori kombinatorial. Sebelumnya, perlu diketahui bahwa dalam konvensi penamaan pada SQL, nama basis data yang valid harus memenuhi aturan-aturan berikut.

1. Karakter pertama berupa alfabet latin (‘a’ sampai ‘z’ atau ‘A’ sampai ‘Z’), simbol tagar (‘#’), simbol at (‘@’), atau simbol garis bawah (‘_’).
2. Karakter selanjutnya berupa alfabet latin (‘a’ sampai ‘z’ atau ‘A’ sampai ‘Z’), simbol tagar (‘#’), simbol at (‘@’), simbol dollar (‘$’), simbol garis bawah (‘_’), atau angka desimal (‘0’ sampai ‘9’).
3. Panjang nama maksimal 128 huruf.

[10]

A. Jumlah Penebakan Maksimum Teknik Brute-Force

Dengan teknik ini, penguji harus mencoba semua kemungkinan kombinasi nama yang valid. Jumlah kombinasi ini dapat dihitung menggunakan kaidah perkalian dan kaidah penjumlahan teori kombinatorial. Untuk karakter pertama, jumlah kemungkinan karakter ada $26 + 26 + 3 = 55$ buah karater, sedangkan untuk karakter yang melanjutkannya terdapat $26 + 26 + 4 + 10 = 66$ buah karakter.

Perhitungan dapat dilakukan dengan cara menggunakan kaidah penjumlahan pada setiap kemungkinan panjang nama, dan menggunakan kaidah perkalian untuk mencari semua kemungkinan nama dengan panjang tertentu. Untuk nama dengan 1 karakter, terdapat 55 kemungkinan. Untuk nama dengan 2 karakter, terdapat 55×66 kemungkinan (kaidah perkalian). Demikian seterusnya dijumlahkan sampai nama dengan panjang 128 huruf. Dengan perhitungan ini, didapatkan kemungkinan kombinasi atau penebakan maksimum sebanyak 6.47×10^{232} buah.

Jumlah karakter yang mungkin	55	66	66	66	…
Posisi	1	2	3	4	…

Gambar 15. Ilustrasi perhitungan jumlah kombinasi nama basis data dalam SQL

B. Jumlah Penebakan Maksimum Teknik Substring

Dengan teknik *substring* penebakan nama dilakukan tidak secara langsung sebagai satu kata namun secara satu karakter per satu karakter. Karena ini, jumlah penebakan maksimum dapat dihitung dengan kaidah penjumlahan dari kemungkinan karakter pada tiap posisi. Perlu diperhatikan bahwa panjang nama dapat ditentukan sepanjang iterasi, karena fungsi *SUBSTRING* akan mengembalikan nilai tidak valid untuk posisi potongan karakter yang lebih besar dari panjang *string*. Kaidah penjumlahan dapat dipakai dengan cara menjumlahkan semua kemungkinan dalam satu posisi karakter. Untuk posisi pertama, terdapat 55 kemungkinan karakter, untuk posisi kedua dan seterusnya terdapat 66 kemungkinan karakter. Dengan kaidah penjumlahan, didapat jumlah penebakan maksimum adalah $55 + 66 \times 127 = 8437$ kali.

C. Jumlah Penebakan Maksimum Teknik Substring dengan Bit Shifting

Dalam teknik ini, nama ditebak secara satu bit per satu bit. Karena semua karakter yang valid dalam nama ini berupa karakter 8-bit, maka untuk satu karakter perlu dilakukan maksimum 8 kali penebakan (satu kali untuk masing-masing posisi bit). Perlu diperhatikan juga bahwa sama seperti teknik sebelumnya, panjang nama dapat ditentukan sepanjang iterasi karena menggunakan fungsi yang sama. Perhitungan maksimum penebakan dapat menggunakan kaidah penjumlahan dengan cara yang sama seperti pada subbab sebelumnya, yaitu menjumlahkan semua kemungkinan karakter. Dengan kaidah penjumlahan didapat penebakan maksimum sebanyak $8 \times 128 = 1024$ kali.

V. Perbandingan Teknik Injeksi SQL dengan Pendekatan Analisis Kompleksitas Algoritma

Selain melalui pendekatan kombinatorial, kemangkusan teknik injeksi SQL berbasis boolean dapat dilihat dengan pendekatan analisis kompleksitas algoritma. Tentunya masing-masing teknik memiliki kompleksitas algoritma yang berbeda-beda. Algoritma yang lebih mangkus memiliki kompleksitas yang ordenya lebih rendah pada spektrum kompleksitas algoritma yang telah dibahas pada bab sebelumnya.. Perlu diperhatikan bahwa nilai n yang digunakan adalah panjang nama yang akan ditebak.

A. Kompleksitas Algoritma Teknik Brute-Force

Menggunakan teknik *brute-force*, tiap kemungkinan kombinasi perlu dicoba. Artinya, diperlukan iterasi masing-masing karakter dengan perulangan bersarang sejumlah karakter yang mungkin pada posisinya, dengan n buah posisi. Sebagai contoh, untuk kemungkinan panjang $n = 2$, perlu diiterasi semua kemungkinan karakter pertama (55) dan karakter kedua (66) dalam perulangan bersarang. Kompleksitas dari algoritma ini dapat dihitung dengan perkalian karena menggunakan perulangan yang bersarang.

Didapatkan kompleksitas waktu $T(n) = 55 \times 66^{n-1}$. Untuk mencari nilai kompleksitas dalam notasi "O-Besar",

$$55 \times 66^{n-1} \leq 66 \times 66^{n-1} = 66^n$$

Untuk $C = 1$ dan $n_0 = 1$, $f(n) = 66^n$, atau kompleksitas algoritma dalam notasi "O-Besar" yaitu $O(f(n)) = O(66^n)$. Algoritma ini termasuk algoritma eksponensial.

B. Kompleksitas Algoritma Teknik Substring

Dengan teknik *substring* penebakan dilakukan satu karakter per satu karakter. Oleh karena itu, tidak seperti teknik *brute-force* di mana dibutuhkan perulangan bersarang untuk masing-masing posisi dalam kunci yang ditebak, dalam teknik ini, algoritma yang digunakan membutuhkan hanya satu perulangan bersarang. Dalam algoritma ini perlu diiterasikan posisi yang akan ditebak dan karakter yang harus ditebak, sketsa kasar dari algoritma ini dapat dilihat pada gambar 16. Dalam sketsa tersebut belum diimplementasikan pencatatan hasil.

```
{Fungsi tebak1(x,y) didefinisikan sebagai
fungsi untuk menebak y sebagai karakter
pada posisi x}

  A = {karakter-karakter posisi pertama}
  B = {karakter-karakter posisi sisanya}
  p = 55 {panjang A}
  q = 66 {panjang B}

  for i<-1 to n do
      if (i=1) then
            for j<-0 to p do
                  tebak1(i, A[j])
      else
            for j<-0 to q do
                  tebak1(j, B[j])
```

Gambar 16. Sketsa kasar algoritma penebakan dalam injeksi SQL berbasis boolean menggunakan teknik *substring*

Dari sketsa di atas, jika kompleksitas waktu dihitung dari pemanggilan fungsi *tebak1*. Untuk $i = 1$, fungsi dieksekusi 55 kali, selebihnya, fungsi diekseskusi 66 kali. Maka, dapat disimpulkan $T(n) = 55 + 66(n-1)$. Untuk mencari nilai kompleksitas dengan notasi "O-Besar",

$$55 + 66(n-1) \leq 66 + 66(n-1) = 66n$$

Untuk $C = 66$ dan $n_0 = 1$, $f(n) = n$, atau kompleksitas algoritma dalam notasi "O-Besar" didapat $O(f(n)) = O(n)$, algoritma ini termasuk linear.

C. Kompleksitas Algoritma Teknik Substring dengan Bit Shifting

Dengan teknik *substring* dengan *bit shifting* penebakan dilakukan satu bit per satu bit dari tiap karakter. Algoritma yang digunakan mirip dengan algoritma pada teknik substring namun lebih sederhana. Sketsa kasar algoritma ini dapat dilihat pada gambar 17. Pencatatan hasil belum diimplementasikan.

```
{Fungsi tebak2(x,y) didefinisikan sebagai
fungsi untuk menebak apakah bit ke-y
karakter pada posisi ke-x adalah 1}

  for i<-1 to n do
      for j<-0 to 7 do
            tebak2(i, j)
```

Gambar 17. Sketsa kasar algoritma penebakan dalam injeksi menggunakan teknik *substring* dengan *bit shifting*

Dari sketsa di atas dapat dihitung kompleksitas dilihat dari pemanggilan fungsi *tebak2*. Untuk setiap iterasi nilai i (terdapat n kali iterasi), dilakukan 8 kali pemanggilan fungsi. Maka, kompleksitas algoritma ini adalah $T(n) = 8n$. Untuk $C = 8$ dan $n_0 = 1$, $f(n) = n$, notasi "O-Besar" untuk algoritma ini adalah $O(n)$. Algoritma ini juga termasuk linear.

VI. KESIMPULAN

Perbandingan ketiga teknik dengan pendekatan teori kombinatorial dan kompleksitas algoritma pada dua bab sebelumnya dapat dirangkum dalam tabel 2 di bawah ini.

Nama Teknik	Penebakan Maksimum	Kompleksitas Algoritma
Brute-Force	6.47×10^{232}	$O(66^n)$, eksponensial
Substring	8437	$O(n)$, linear
Substring dengan Bit-Shifting	1024	$O(n)$, linear

Tabel 2. Perbandingan masing-masing teknik

Dapat disimpulkan, dari ketiga teknik yang dibandingkan, teknik yang paling paling efektif adalah teknik injeksi SQL berbasis boolean menggunakan *substring* dengan *bit-shifting* karena memerlukan penebakan maksimum yang paling sedikit dan kompleksitas algoritma yang ordenya terendah di spektrum.

VII. UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa karena hanya atas berkat-Nya saja penulis dapat menyelesaikan makalah dengan judul "Aplikasi Teori Kombinatorial dan Kompleksitas Algoritma dalam Perbandingan Teknik Injeksi SQL Berbasis Boolean" ini tepat waktu. Selain itu, penulis juga mengucapkan terima kasih kepada para dosen pengampu mata kuliah IF2120 Matematika Diskrit, pada Semester I Tahun 2019/2020, Ibu Fariska Zakhralativa, Bapak Rinaldi Munir, serta Ibu Harlili atas ilmu yang diberikan.

DAFTAR PUSTAKA

[1] https://portswigger.net/web-security/sql-injection, diakses pada 28 November 2019.
[2] https://portswigger.net/web-security/sql-injection/blind, diakses pada 28 November 2019.
[3] https://www.securityidiots.com/Web-Pentest/SQL-Injection/Blind-SQL-Injection.html, diakses pada 29 November 2019.
[4] https://www.exploit-db.com/papers/17073, diakses pada 29 November 2019.
[5] http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Teori%20Bilangan%20(2015).pdf, diakses pada 30 November 2019.
[6] http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Kompleksitas%20Algoritma%20(2015).pdf, diakses pada 30 Novemer 2019.
[7] https://dev.mysql.com/doc/refman/8.0/en/select.html, diakses pada 1 Desember 2019.
[8] https://www.oracle.com/database/what-is-a-relational-database/, diakses pada 1 Desember 2019.
[9] https://dev.mysql.com/doc/refman/8.0/en/select.html, diakses pada 1 Desember 2019.
[10] https://docs.microsoft.com/en-us/sql/relational-databases/databases/database-identifiers?view=sql-server-2017, diakses pada 1 Desember 2019.

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, 3 Desember 2019

Steve Bezalel Iman Gustaman, 13518018