Penerapan Rekursif dan Analisa Kompleksitas Algoritma Pada Brute Force Login

Aryya Dwisatya Widigdha/13512043
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13512043@std.stei.itb.ac.id

***Abstrak*—Makalah ini membahas tentang rekursifitas dan kompleksitas algoritma pada brute force login. Selain itu, makalah ini juga membahas algoritma brute force login secara mendetil sebagai pengetahuan untuk para admin website agar websitenya aman dari serangan *brute force* login karena pada kenyataannya saat ini banyak sekali website yang rentan akan teknik penyerangan ini. Makalah ini bertujuan untuk mengurangi korban penyerangan menggunakan algoritma *brute force* hacker dan memotivasi pembaca pada umumnya dan admin website pada khususnya untuk lebih memperhatikan keamanan situsnya serta untuk berinovasi dalam hal keamanan terhadap suatu teknik penyerangan salah satunya *brute force login*.**

***Kata kunci*—rekursif, kompleksitas, *brute force login*, password**

I. PENDAHULUAN

Pada era digital seperti sekarang ini, berbagai informasi banyak yang disimpan pada website. Hampir seluruh informasi bisa kita temukan di internet. Namun ternyata, tidak semua informasi tersebut ditujukan untuk umum melainkan ada informasi yang ditujukan untuk kalangan tertentu saja.

Biasanya, untuk mengakses informasi tersebut, dibutuhkan sandi lewat. Walaupun informasi tersebut tidak seluruhnya dapat diakses oleh publik dan diperlukan sandi lewat untuk mengaksesnya, tapi tetap saja ada kemungkinan informasi dapat diakses dan digunakan oleh orang yang tidak berhak.

Para *attacker* (penyerang) situs bisa mendapatkan berbagai informasi yang tersimpan di dalam web meskipun tidak memiliki akses yang sah. Salah satu cara yang dapat digunakan untuk melakukan hal tersebut adalah dengan menggunakan teknik *brute force login.*

II. PENJELASAN MENGENAI REKURSIF

1. Definisi dan contoh rekursif

Suatu objek dikatakan rekursif jika ketika didefinisikan, ia tetap ada dalam pendefinisiannya. Contohnya, bahagia adalah ketika kita melihat orang yang kita sayang bahagia. Bahagia masih terdapat dalam definisinya sehingga bahagia dikatakan rekursif.

Dalam penerapannya, rekursi didefinisikan sebagai fungsi sehingga ketika dilakukan pemanggilan, rekursif dapat mengembalikan nilai dengan masukan yang dapat diubah. Ada dua bagian penting yang harus ada pada suatu fungsi rekursif yakni basis dan rekurens.

Basis merupakan bagian yang berisi nilai fungsi yang terdefinisi secara eksplisit atau jelas dan berfungsi untuk menghentikan rekursif. Sedangkan rekurens merupakan bagian yang mendefinisikan fungsi pada dalam dirinya sendiri atau sederhananya merupakan pemanggil dirinya sendiri. Tentu nilai masukan pada bagian ini selalu mendekati basis supaya fungsi dapat berhenti.

Contoh fungsi yang rekursif

```
function faktorial($n){
    if($n==0){
        return 1;
    }else{
        return $n*faktorial($n-1);
    }
}
```

Gambar 1- Fungsi rekursif faktorial

Fungsi tersebut merupakan fungsi untuk menghitung factorial. Fungsi tersebut ditulis dalam bahasa pemrograman php.

Ketika fungsi tersebut dijalankan menggunakan perintah factorial(3) maka fungsi tersebut akan menghasilkan solusi sebagai berikut:

= 3 * factorial(3-1)
= 3 * 2 * factorial(2-1)
= 3 * 2 * 1 * factorial(1-1)
= 3 * 2 * 1 * 1
=6

```
    if($n==0){
        return 1;
```

Gambar 2- basis

Merupakan basis karena berisi nilai keluaran yang eksplisit dan menghentikan fungsi, tidak memanggil dirinya sendiri.

```
}else{
    return $n*faktorial($n-1);
}
```

Gambar 3- rekurens

Merupakan rekurens karena memanggil dirinya sendiri dengan masukan yang semakin mendekati basis.

III. PENJELASAN MENGENAI KOMPLEKSITAS ALGORITMA

Hampir seluruh algoritma memiliki berbagai alternatif solusi. Namun, tentu dari berbagai solusi tersebut ada solusi terbaik yakni dengan kemangkusan ditinjau dari sisi ruang memori maupun waktu eksekusi. Jadi, kompleksitas algoritma merupakan kemangkusan suatu algoritma ditinjau dari sisii waktu eksekusi maupun kebutuhan ruang memori.

Kemangkusan ruang memori maupun waktu ditentukan oleh masukan yang diterima oleh program. Dari kemangkusan tersebut, dapat dinilai algoritma mana yang paling cocok untuk menyelesaikan suatu masalah.

a. Masukan
   Masukan merupakan suatu nilai yang diterima oleh suatu algoritma untuk diproses. Dalam kompleksitas algoritma, masukan dinotasikan dengan huruf n.
   Contoh:
   - Algoritma pengurutan 1000 elemen larik maka n = 1000
   - Algoritma perkalian dua buah matriks MxM maka n = 2m
   - Algoritma perkalian konstanta dengan array sejumlah c maka n= 2c
   - Dan lain-lain.

b. Kompleksitas Waktu
   Kompleksitas waktu T(n) merupakan kompleksitas algoritma dengan meninjau sisi waktu dari suatu algoritma. Kompleksitas waktu merupakan kuantisasi dari tahapan komputasi yang dibutuhkan untuk menyelesaikan algoritma sebagai fungsi dengan sejumlah masukan.
   Walaupun dalam suatu algoritma terdapat berbagai macam operasi seperti baca/tulis, aritmetika, pengisian nilai, pengaksesan elemen larik, dan lain-lain, tapi operasi yang menjadi dasar perhitungan suatu algoritma hanya operasi yang mendasari algoritma tersebut saja.
   Contoh: Algoritma menghitung jumlah total elemen larik a.
   Sum ← 0
   For I ← 1 to n do
       Sum ← Sum + a[i]
   Endfor
   Pada algoritma tersebut, operasi yang mendasarinya adalah operasi Sum ← Sum + a[i]. Oleh karena itu, operasi itu saja yang dihitung sebagai kompleksitas waktu dari algoritma tersebut. Berdasarkan kenyataan tersebut diketahui bahwa kompleksitas waktu algoritma menghitung jumlah total elemen larik a adalah T(n) = n atau dapat dibaca sebesar jumlah masukan yang diterima oleh program.

   Klasifikasi Kompleksitas Waktu
   i. $T_{max}(n)$
      $T_{max}(n)$ merupakan suatu nilai yang merepresentasikan kompleksitas waktu untuk kasus terburuk atau dengan kata lain waktu yang maksimal yang dibutuhkan untuk menyelesaikan algoritma dalam memproses masukan sejumlah n.
   ii. $T_{min}(n)$
      Berbeda dengan $T_{max}(n)$, $T_{min}(n)$ merepresentasikan kompleksitas waktu untuk kasus terbaik atau dalam kata lain $T_{min}(n)$ merepresentasikan kebutuhan waktu minimum suatu algoritma dalam memproses masukan sejumlah n.
   iii. $T_{avg}(n)$
      Tentu, dari nilai maksimal dan minimal yang ada pada kompleksitas waktu, ada nilai yang merepresentasikan nilai rata-rata waktu yang dibutuhkan. Kebutuhan waktu rata-rata yang dibutuhkan untuk memproses suatu masukan disebut $T_{avg}(n)$

c. Kompleksitas ruang
   Berbeda dengan kompleksitas waktu T(n) yang meninjau suatu algoritma dari sisi waktu, kompleksitas ruang meninjau suatu algoritma dari ruang memori yang digunakan untuk memproses sejumlah masukan.

d. Big-O
   Dalam kompleksitas waktu algoritma, terdapat pertumbuhan mengikuti persamaan yang merepresentasikan masukan. Sebagai contoh terdapat kompleksitas waktu $T(n) = 2n^2+6n+1$.

n	$T(n) = 2n^2+6n+1$	n^2
10	261	100
100	2061	1000
1000	2006001	1000000
10000	2000060001	1000000000

Berdasarkan tabel tersebut nampak bahwa kompleksitas waktu tumbuh mengikuti nilai n2. Oleh karena itu T(n) dikatakan berorde n^2 dan dituliskan dalam notasi $T(n) = O(n^2)$.

O 'Big-O' merupakan notasi kompleksitas waktu asimtotik. T(n) = O(f(n)) artinya T(n) berorde paling besar f(n)) bila terdapat konstanta C dan n_0 sedemikian sehingga T(n) <= C(f(n)) untuk n >= n_0 yang mana f(n) adalah batas lebih atas dari T(n) untuk n yang besar.

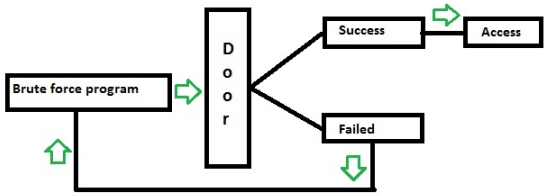
IV. PENJELASAN MENGENAI *BRUTE FORCE*

Berbicara mengenai website, tentu tidak lepas dari pikiran seorang attacker untuk mendapatkan akses lebih

daripada publik pada website tersebut. Salah satu cara untuk mendapatkan akses tersebut adalah dengan *brute force login*.

i. Definisi
   Brute force merupakan suatu algoritma dengan metode *trial and error* yang dilakukan secara terus menerus hingga ditemukan satu solusi yang benar. Algoritma ini dapat menjadi solusi untuk menerobos suatu sistem *login*. Tentu, ada suatu kelemahan yang menjadi ciri khas dari algoritma ini yakni kompleksitas waktu yang besar karena algoritma ini akan mencoba setiap kemungkinan yang ada hingga benar-benar ditemukan solusi yang memenuhi baik di awal, tengah, maupun akhir kemungkinan solusi.

ii. Cara kerja
   Brute force memiliki cara kerja yang sederhana yakni mengecek semua kemungkinan solusi yang ada pada lapisan pengecekan. Bila ditemukan jawaban maka proses akan berhenti, tapi ketika solusi yang ditawarkan tidak memenuhi maka akan dilakukan pengecekan menggunakan solusi selanjutnya hingga ditemukan solusi yang tepat atau ditemukan kondisi berhenti yang lain.



Gambar 4 - Alur kerja brute force

Alur kerja atau cara kerja yang demikian sebenarnya mirip dengan rekursif yang memiliki basis dan rekurens. Dapat dilihat pada gambar 4 bahwa program brute force akan memberikan sebuah solusi untuk dilakukan pengecekan oleh door. Bila door menerima solusi tersebut sebagai solusi yang tepat maka program akan berhenti, tapi bila solusi tidak diterima oleh door maka pengecekan akan dilakukan terus menerus.

V. REKURSIF DAN KOMPLEKSITAS ALGORITMA PADA *BRUTE FORCE LOGIN*

Brute force, kompleksitas algoritma, dan rekursif memiliki suatu keterkaitan terlebih ketika membahas tentang *brute force login*. *Brute force login* adalah suatu cara untuk mendapatkan akses *private* secara tidak sah menggunakan algoritma *brute force* untuk menjebol sandi lewat yang ada.

Sebagai contoh kasus, diberikan suatu algoritma *brute force login* sebagai berikut.

```
function bruteforce (Csource: array of string
sejumlah, CNum :  integer) → CC

Kamus
CC : string

Algoritma
CC ← Csource[CNum-]

if CNum > 0 then
        if (proses(CC)) then
                bruteforce(Csource,CNum-1)
        else
                → CC
else
        {do nothing}
```

Pada algoritma di atas, diasumsikan fungsi *proses(CC)* sudah diketahui realisasinya yakni mengirimkan CC kepada server untuk selanjutnya menerima respon dari server berupa jawaban apakah solusi yang ditawarkan tepat atau tidak.

Untuk menghitung kompleksitas waktu pada algoritma tersebut, dipakai *proses(CC)* sebagai operator acuan. Oleh karenanya, dapat diketahui bahwa kompleksitas waktu dari algoritma brute force login tersebut memiliki tiga macam kemungkinan yakni $T_{max}(n) = n$, $T_{min}(n) = 1$, atau $T_{avg}(n) = (n+1)/2$. $T_{max}(n) = n$ karena dimungkinkan solusi yang tepat berada pada awal *array of string* sehingga diperlukan waktu penuh untuk mencoba seluruh elemen *array* hingga ditemukan suatu solusi yang pas. Sedangkan $T_{min}(n) = 1$ karena dimungkinkan solusi tepat berada di akhir *array* sehingga ketika pertama kali melakukan pengecekan, solusi yang tepat langsung ditemukan dengan waktu yang minimal. Sayangnya, kedua kasus tersebut khusus dan memiliki kemungkinan 100/CNum saja. Kasus yang paling umum terjadi adalah kasus rata-rata di maka solusi bisa berada pada elemen kedua *array*, elemen ketujuh *array* ataupun elemen lain yang bukan elemen pertama maupun terakhir. Oleh karena itulah diambil nilai rata-rata tepat di tengah *array* sebagai representasi kasus umum kompleksitas algoritma brute force login.

Untuk lebih menguatkan algoritma tersebut, berikut adalah realisasi algoritma brute force tersebut dalam bahasa php

```
<?php
/* file rekursif.php */
function proses($CC){
        if(preg_match("/Login salah/",
                file_get_contents
        ("http://localhost/xb/lindex.php?password=
".$CC)))
        {
                return true;}
{               return false;
        }
```

```
/* file rekursif.php */
function bruteforce($data,$CNum){
	if($CNum>0){
		if (proses($data[$CNum-1])){
			bruteforce($data,$CNum-
1);
		}else{
			print $data[$CNum-1];
		}
	}
}

$listkata =
array("MatdsisA","kemana","MsatdisA","aku","dia"
,"kamu","kesmana");

bruteforce($listkata,count($listkata));

?>
```

```
<?php
/* lindex.php */
	if($_GET['password'] != "MatdisA"){
		print "Login salah";
	}
?>
```

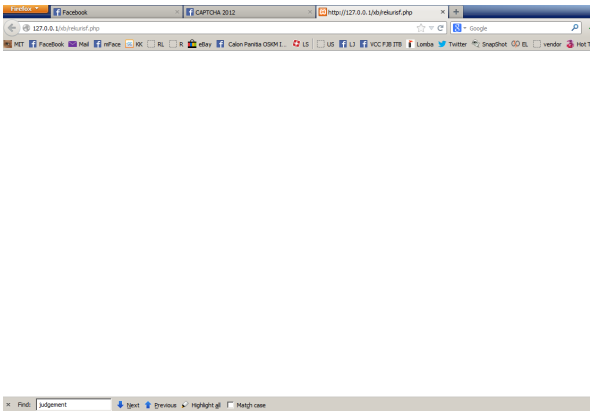
File rekurisf.php merupakan file utama yang mengandung algoritma brute for untuk menembus fie lindex.php. Asumsikan attacker telah menganalisis file lindex.php sehingga attacker mengetahui bahwa data yang diterima oleh lindex.php disalurkan melalui *method GET*.

Sandi lewat yang tepat untuk menembus lindex.php adalah MatdisA. File Rekursif.php akan bekerja terus menerus hingga sandi lewat atau solusi yang tepat ditemukan ataupun ketika sudah tidak ada kemungkinan sandi lewat yang dapat digunakan.

```
$listkata =
array("MatdsisA","kemana","MsatdisA","aku","dia","
kamu","kesmana");
```

$listkata merupakan *array of string* sebagai masukan untuk fungsi brute force yang mana akan diproses oleh fungsi *proses*. Fungsi *proses* akan mengecek respon dari server, apakah server memberikan pemberitahuan bahwa sandi lewat yang dikirim salah. Bila salah, maka rekursifitas akan berlanjut dengan nilai CNum yang merupakan jumlah kemungkinan sisa yang makin mengecil menuju basis.

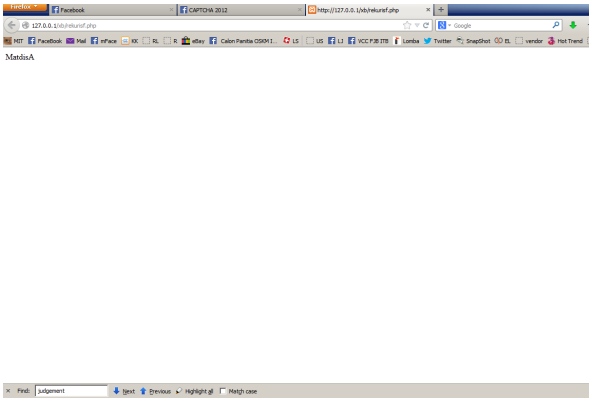
Ketika file rekursif.php diekesusi maka akan menghasilkan halaman kosong.



Gambar 5 - Hasil eksekusi file rekursif.php

Hal ini terjadi karena sandi lewat yang tepat tidak ada pada *array* $listkata. Namun, ketika sandi lewat yang tepat berada pada *array* $listkata maka seketika ketika sandi lewat ditemukan, ia akan ditampilkan.

```
$listkata =
array("MatdisA","kemana","MsatdisA","aku","dia","
kamu","kesmana");
```



Gambar 6 - Sandi lewat ditemukan

Bila dieksekusi, memang tidak terlalu nampak perbedaan waktu eksekusi ketika ada maupun tidak ada sandi lewat yang tepat. Hal ini terjadi karena kompleksitas waktu algoritma tersebut berorde n sehingga dibutuhkan data yang banyak untuk memberikan perbedaan yang signifikan.

VI. TIPS AMAN DARI BRUTE FORCE LOGIN

Ada beberapa tips agar website kita aman dari kerentanan terhadap brute force login yakni

i. Gunakan huruf besar, angka, dan tanda baca pada password yang dimiliki. Hal ini bertujuan untuk meminimalisasi kemungkinan password Anda tertebak karena sesungguhnya bruteforce merupakan *dictionary attack* yang bertumpu pada daftar password yang umum digunakan.
ii. Gunakan enkripsi terhadap password Anda.

Enkripsi penting digunakan agar password yang Anda miliki tidak sepenuhnya dapat terlihat jelas oleh *attacker*. Enkripsi yang dapat digunakan antara lain SHA1, MD5, maupun base64.

iii. Jadilah Anti-mainstream. Ketika Anda menjadi seorang admin website, usahakan halaman admin anda tidak berada pada halaman atau folder umum seperti admin.php, /admin/index.php, adminpage.php dan sebagainya karena hal tersebut memudahkan *attacker* untuk menentukan titik awal penyerangan.

iv. Gunakan *case-sensitive* pada saat pengecekan password. Hal ini sangat dapat mereduksi kemungkinan password anda dengan mudah tertebak atau ditembus menggunakan brute force login karena Aku dan aKu merupakan dua hal yang berbeda ketika Anda menggunakan *case-sensitive*.

v. Catat IP address pengakses. Paling tidak, dengan mengetahui bahwa Anda mencatat setiap IP address pengakses halaman tersebut, *attacker* akan berpikir ulang untuk melakukan aksinya.

VII. KESIMPULAN

Berdasarkan percobaan dan perhitungan yang penulis lakukan dapat disimpulkan bahwa:

i. Brute force login merupakan salah satu cara mendapatkan akses informasi, tapi bukan cara yang terbaik karena memiliki kemungkinan untuk tereksekusi dalam waktu n maupun (n+1)/2 satuan.

ii. Rekursifitas dapat mempersingkat suatu algoritma.

iii. Kompleksitas waktu dapat dijadikan dasar dalam menentukan algoritma terbaik yang harus digunakan dan memperkirakan alur kerja suatu algoritma agar dicapai kompleksitas waktu minimal.

REFERENSI

[1] http://www.w3schools.com/php/php_arrays.asp. Diakses pada 15 Desember 2013 pukul 00.59

[2] http://www.php.net/manual/en/language.types.array.php. Diakses pada 15 Desember 2013 pukul 00.59

[3] http://cplus.about.com/od/glossar1/g/bruteforce.htm. Diakses pada 14 Desember 2013 pukul 23.15

[4] http://www.techopedia.com/definition/18091/brute-force-attack . Diakses pada 14 Desember 2013 pukul 23.00

[5] *Munir,Rinaldi. Diktat Kuliah Matematika Diskrit* Diskrit Edisi Keempat.Bandung: Program Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika 2006.

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, 27 November 2013

Aryya Dwisatya Widigdha
13512043