Meningkatkan Keamanan Kata Sandi dengan Memahami Kompleksitas Algoritma Serangan Brute Force

Maria Flora Renata Siringoringo - 13522010[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]author@itb.ac.id

***Abstract*—Kemanan kata sandi adalah hal yang masih sering diabaikan. Padahal, dalam kehidupan sehari-hari kita sangat sering menggunakan kata sandi untuk autentikasi. Kata sandi yang terlalu lemah dapat dengan mudah diretas menggunakan serangan sederhana seperti serangan *brute force.* Memahami cara kerja sebuah serangan dapat membuat kita lebih sadar akan cara membuat kata sandi yang lebih aman.**

***Keywords— Bits of entropy, brute force,* kata sandi, kompleksitas algoritma**

I. PENDAHULUAN

Dalam kehidupan sehari-hari, tidak jarang kita menemukan sistem autentikasi menggunakan kata sandi. Hampir semua perangkat lunak yang menggunakan sistem akun, menerapkan kata sandi. Rata-rata dari jumlah kata sandi yang dimiliki seseorang adalah 100 kata sandi.

Walaupun kata sandi sangat sering digunakan, banyak orang tidak sadar tentang pentingnya keamanan kata sandi. Sebuah riset menunjukkan bahwa kata sandi yang paling sering digunakan di seluruh dunia adalah "123456". Kata sandi ini membutuhkan waktu kurang dari satu detik untuk dipecahkan oleh seorang peretas. Dari 200 kata sandi paling populer, 82.5% memerlukan waktu kurang dari satu menit untuk dipecahkan.

Kata sandi yang terlalu lemah akan membuat pemiliknya sangat rentan terhadap peretasan. Seorang peretas dapat dengan mudah mendapakan kata sandi orang tersebut dan mengakses akunnya di berbagai perangkat lunak. Seorang peretas dapat menyebabkan kerugian yang besar bagi korbannya, dalam bentuk pencurian data diri, memasukkan *malware*, dan merubah sebuah situs sepenuhnya.

Untuk itu, makalah ini bertujuan untuk meningkatkan pemahaman tentang keamanan kata sandi dengan menggunakan analisa kompleksitas algoritma terhadap salah satu metode peretasan kata sandi yang paling sering digunakan, yaitu serangan *brute force*. Dengan mempelajari algoritma yang digunakan dalam peretasan kata dapat ditentukan gambaran perbandingan panjang kata sandi dengan keamanannya, serta kriteria lain yang membuat kata sandi menjadi cukup aman.

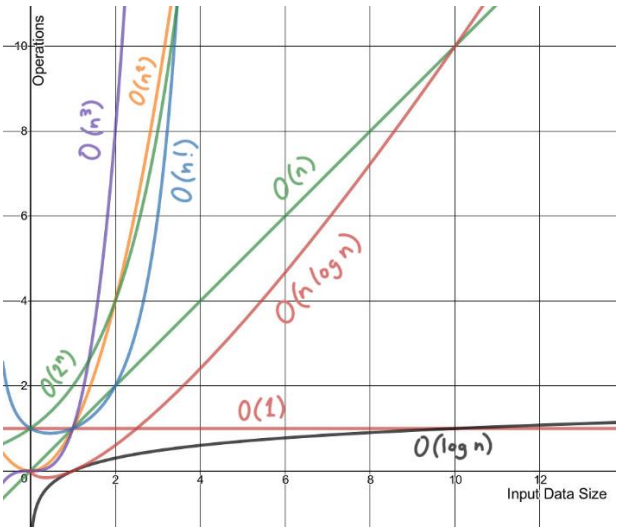
II. KOMPLEKSITAS ALGORITMA

Kompleksitas algoritma adalah ukuran kemangkusan sebuah algoritma ditinjau dari segi waktu dan ruang. Ukuran ini hanya didasarkan pada jumlah input dan bukan pada data asli waktu atau ruang yang diperlukan. Hal tersebut dilakukan supaya hasil perhitungan kompleksitas algoritma tersebut universal, tidak bergantung pada mesin dan bahasa yang digunakan. Kompleksitas algoritma membandingkan beberapa algoritma sebagai susunan langkah, sehingga mempermudah pemilihan algoritma yang tepat untuk sebuah program.

Dalam pengukuran kompleksitas waktu sebuah algoritma, hanya instruksi dasar atau instruksi khas algoritma tersebut yang dipertimbangkan, misalnya operasi pertukaran dan perbandingan pada algoritma pengurutan dan operasi penjumlahan dalam algoritma menghitung ukuran larik. Jumlah instruksi khas yang dilakukan dalam skenario terburuk (*worst case scenario*) akan membentuk sebuah fungsi dalam n yang disebut T(n).

Jika T(n) memiliki beberapa suku, suku yang bernilai lebih besar akan mendominasi pertumbuhan T(n). Konstanta yang ada pada T(n) juga dapat menjadi berbeda jika dihitung dari bahasa pemrograman yang berbeda. Kompleksitas waktu yang dihitung secara asimptotik hanya mengambil suku terbesar dan menghilangkan konstantanya, sehingga menjadi lebih sederhana.

Notasi kompleksitas algoritma yang paling sering digunakan adalah notasi O-Besar atau Big-O notation. notasi ini menghitung kompleksitas algoritma secara asimptotik, dan didefinisikan sebagai batas atas kompleksitas sebuah algoritma. Batas atas yang diambil harus mendekati kompleksitas asli algoritma tersebut supaya hasil perhitungan lebih akurat. Batas atas sebuah algoritma lebih mudah dihitung dibandingkan dengan batas aslinya, contohnya sebuah algoritma yang melakukan pengulangan sejumlah n, lalu (n-1), lalu (n-2), hingga 1 memiliki batas atas n^2.



Gambar 1: Grafik pertumbuhan jumlah operasi pada ukuran notasi Big-O yang berbeda.
(Sumber: https://victoria.dev/blog/a-coffee-break-introduction-to-time-complexity-of-algorithms/)

Notasi Big-Omega juga merupakan notasi kompleksitas waktu asimptotik. Big-Omega tidak menghitung batas atas, melainkan batas bawah dari *worst case scenario* sebuah algoritma. Notasi Big-Theta adalah notasi yang juga asimptotik. Big-Theta mengukur batas atas dan batas bawah algoritma, sehingga Big-Thetha menunjukkan kompleksitas waktu rata-rata sebuah algoritma.

Menggunakan analisa berbasis kompleksitas algoritma ini, dapat ditentukan seberapa besar pertumbuhan waktu eksekusi program berhubungan dengan besarnya input atau kata sandi pengguna.

III. KATA SANDI

Kata sandi untuk komputer pertama kali dicetuskan oleh Fernando Corbató. Saat itu, kata sandi tersebut digunakan di komputer bersama MIT untuk menjaga privasi dokumen di dalam komputer tersebut.

Sekarang, autentikasi dengan kata sandi sudah menjadi standar untuk perangkat lunak yang memiliki sistem akun. Untuk mengakses sebuah akun, pengguna harus memasukkan sebuah pengenal, biasanya dalam bentuk *username* atau email, dan memasukkan kata sandinya. Kedua input tersebut akan dibandingkan dengan *database* pengguna yang dimiliki oleh sistem, lalu pengguna akan masuk jika ditemukan pasangan yang sesuai.

Beberapa kelebihan dari penggunaan kata sandi adalah:

1. Pengguna sudah mengenal sistem kata sandi
2. Autentikasi menggunakan kata sandi sangat sederhana dan mudah dilakukan, umumnya hanya perlu mengingat kata sandi sendiri dan tidak memerlukan barang fisik.
3. Membebaskan pengguna untuk mengatur kata sandinya sendiri, termasuk seberapa sering mengganti kata sandi.

Beberapa dari kelebihan tersebut sebenarnya menimbulkan masalah sendiri. Sistem kata sandi digunakan oleh hamper setiap perangkat lunak yang digunakan sehari-hari, sehingga seorang pengguna merasa lebih nyaman untuk memakai kata sandi dan pengenal yang sama untuk semua akunnya. Pengguna tersebut akan sangat rentan jika salah satu akunnya sudah diretas.

Daftar kata sandi pengguna awalnya sering disimpan dalam bentuk *plain text* secara langsung. Akibatnya, seseorang dapat mengambil data tersebut dengan mudah. Sebuah perusahaan bernama RockYou pernah menyimpan kata sandi dalam *plain text*. Perusahaan tersebut diretas pada 2009 dan para peretasnya mengambil dan mempublikasikan daftar kata sandi seluruh pengguna RockYou. Untuk mencegah hal serupa, daftar kata sandi biasanya disimpan dalam bentuk terenkripsi menggunakan fungsi *hash.*

IV. SERANGAN BRUTE FORCE DAN KEAMANAN KATA SANDI

Brute force berarti menggunakan kekuatan besar tanpa memperhatikan efisiensi dan presisi. Serangan *brute force* adalah salah satu jenis serangan yang dapat dilakukan untuk meretas sebuah akun. Dalam serangan *brute force*, peretas sudah memiliki pengenal akun seperti *username*, lalu peretas akan mencoba semua kombinasi kata sandi sampai dia berhasil masuk ke akun tersebut.

```
def mainloop(password,starttime):
    chars=string.printable
    for i in range(1,13):
        print("\nMencoba kata sandi dengan panjang",i,"karakter...")
        lap=time()
        print('Waktu: %.2f detik' % (lap - starttime))
        generator=product(chars,repeat=i)
        test=bruteloop(password,generator)
        if (test!=False):
            return test
```

Gambar 2: Contoh algoritma brute force untuk kata sandi
(Sumber: arsip penulis)

```
def bruteloop(password, generator):
    for c in generator:
        if ''.join(c) == password:
            print('\nKata sandi:', ''.join(c))
            return ''.join(c)
    return False
```

Gambar 3: Fungsi pengecekan kata sandi
(Sumber: arsip penulis)

Algoritma *brute force* di atas akan membuat semua kombinasi kata sandi dari semua karakter *printable* ASCII sepanjang satu karakter sampai kata sandi yang benar ditemukan. Panjang maksimal kata sandi di contoh tersebut adalah 12 karakter.

Fungsi yang digunakan untuk membentuk kombinasi karakter, fungsi product, setara dengan fungsi berikut untuk kombinasi dengan panjang dua karakter:

```
def product(printables):
    result=[]
    for letter1 in printables:
        for letter2 in printables:
            result.append(letter1+letter2)
    return result
```

Gambar 4: Gambaran fungsi product
(Sumber: arsip penulis)

Untuk kombinasi sepanjang satu karakter, algoritma tersebut akan langsung mengembalikan *string* printables dalam bentuk larik. Dalam kasus tersebut, jika hanya menghitung operasi $letter1 + letter2$, algoritma membutuhkan waktu O(n), dengan $n = 94$, sesuai jumlah karakter ASCII yang dapat tercetak.

Kombinasi dengan panjang lebih dari dua karakter harus melakukan fungsi product beberapa kali. Untuk tiga karakter, fungsi akan serupa dengan product(product(printables)) yang berarti hasil dari fungsi akan dikalikan kembali dengan semua karakter ASCII yang dapat tercetak (fungsi yang ditunjukkan pada gambar 4 hanya sebagai gambaran dan tidak dapat melakukan hal tersebut).

Kombinasi dengan panjang dua karakter membutuhkan waktu $O(n^2)$. Kombinasi dengan tiga karakter perlu menjalankan fungsi product dua kali, sehingga memerlukan waktu sebanyak $O(n^4)$. Kombinasi dengan lebih dari tiga karakter memerlukan penggunaan fungsi product lebih banyak, sehingga untuk kombinasi dengan panjang kata kunci adalah N, $O(n^{2(N-1)})$. Untuk algoritma ini, setiap karakter tambahan akan menambah jumlah langkah yang diperlukan sebanyak n^2.

Secara lebih umum, perhitungan keamanan sebuah kata sandi menggunakan konsep *bits of entropy*. *Bits of entropy* dapat diartikan sebagai jumlah tebakan maksimal yang diperlukan untuk meretas kata sandi dengan asumsi peretas mengetahui kumpulan simbol yang digunakan.

$$E = L \times \log_2 R$$

Dengan E sebagai entropi, L sebagai panjang kata sandi, dan R sebagai jumlah karakter yang mungkin digunakan (jumlah anggota *set* symbol yang digunakan).

Dari rumus tersebut, entropi sebuah kata sandi bergantung pada panjangnya dan jenis karakter yang digunakan. Artinya, semakin panjang dan semakin banyak jenis karakter yang dipakai oleh sebuah kata sandi, semakin aman kata sandi tersebut. Batas bawah untuk kata sandi yang aman adalah 75 bit entropi.

Algoritma *brute* force sangat tidak efisien untuk kata sandi yang panjang. Dalam sebuah peretasan asli, algoritma yang digunakan tidak hanya menggunakan *brute force*. Biasanya, serangan digabungkan dengan *dictionary attack* untuk mempersingkat durasi yang dibutuhkan. *Dictionary attack* berarti peretas akan mencoba semua kata di dalam sebuah kamus terlebih dahulu, lalu mencobanya dengan mensubstitusi beberapa huruf menjadi leet talk ato substitusi lain yang sering digunakan. Setelah mencoba menggunakan kamus dan gagal, peretas baru menggunakan *brute force*.

Maka, kata sandi dengan entropi yang tinggi saja tidak menjamin keamanannya. Kata sandi akan lebih aman jika menggunakan kata-kata yang tidak biasa. Kamus yang sering digunakan meliputi daftar kata sandi yang sering digunakan, daftar nama yang sering dimiliki, kata-kata terkait film/televisi, dan berbagai daftar kata lain.

Sebuah riset mempublikasikan 200 kata sandi yang paling sering digunakan di 2023. Kebanyakan kata sandi di dalam daftar tersebut memiliki nilai entropi sangat kecil dan dapat diretas dalam waktu kurang dari satu menit.

Tabel 1: Perhitungan entropi dari sepuluh kata sandi paling populer

Peringkat	Kata Sandi	L	R	Entropi
1	123456	6	10	19.93
2	admin	5	26	23.50
3	12345678	8	10	26.58
4	123456789	9	10	29.90
5	1234	4	10	13.29
6	12345	5	10	16.61
7	password	8	26	37.60
8	123	3	10	9.96
9	Aa123456	8	62	47.63
10	1234567890	10	10	33.22

Nilai entropi sepuluh kata sandi tersebut jauh lebih rendah dari nilai yang disarankan, yaitu 75. Selain itu, kata sandi tersebut dapat dengan mudah ditemukan di dalam sebuah kamus.

Riset lain oleh Moshe Zviran dan William J. Haga pada 1999 menunjukkan bahwa kebanyakan orang menggunakan informasi yang menurutnya penting sebagai kata sandi. Sebanyak 78.4% dari 997 responden menyatakan bahwa mereka menggunakan hal seperti Namanya sendiri, nama orang-orang terdekat dan hewan peliharaan, atau tanggal lahir. Penggunaan informasi seperti ini dapat mempersempir jumlah tebakan yang harus dilakukan oleh seorang peretas.

V. PERCOBAAN

Untuk makalah ini, telah dibuat sebuah program *brute force* sederhana dengan fungsi yang sudah ditunjukkan pada gambar 2, gambar 3, dan gambar 4. Untuk percobaan ini digunakan sebuah laptop dengan spesifikasi sebagai berikut:

- Processor: AMD Ryzen 7 5800H
- GPU: NVIDIA GeForce RTX 3060 6GB
- RAM: 16 GB

Laptop dalam keadaan terhubung dengan sumber listrik saat percobaan dilakukan.

Program yang digunakan adalah sebagai berikut:

```
# MAIN
p=input("Masukkan kata sandi yang akan di brute force ")
start = time()
mainloop(p,start)
end = time()
print('Waktu yang diperlukan: %.2f detik' % (end - start))
```

Gambar 5: Program utama
(Sumber: arsip penulis)

Saat dijalankan, program akan meminta masukan sebuah kata sandi dengan panjang maksimal 12 karakter. Kemudian, program akan membentuk sebuah sekuens yang terdiri dari

seluruh karakter yang dapat diketik oleh pengguna.

Lalu program akan memanggil fungsi mainloop. Di dalam fungsi tersebut, program akan memanggil fungsi product dari itertools untuk membuat produk kartesian dari *string* chars yang berisi seluruh karakter yang dapat diketik oleh pengguna, yaitu: chars = “0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~”

Produk kartesian yang dibentuk akan memiliki panjang satu karakter sampai duabelas karakter. Fungsi mainloop akan memanggil fungsi bruteloop untuk melakukan perbandingan antara kata sandi yang dimasukkan oleh pengguna dengan seluruh isi larik hasil produk kartesian chars. Jika didapatkan hasil yang sama, program akan keluar dari kedua fungsi dan menunjukkan kata sandi yang dipecahkan, serta waktu berjalannya program.

```
Masukkan kata sandi yang akan di brute force a

Mencoba kata sandi dengan panjang 1 karakter...
Waktu: 0.00 detik

Kata sandi: a
Waktu yang diperlukan: 0.00 detik
```

Gambar 6: Keluaran program dengan input a
(Sumber: arsip penulis)

```
Masukkan kata sandi yang akan di brute force aa

Mencoba kata sandi dengan panjang 1 karakter...
Waktu: 0.00 detik

Mencoba kata sandi dengan panjang 2 karakter...
Waktu: 0.00 detik

Kata sandi: aa
Waktu yang diperlukan: 0.00 detik
```

Gambar 7: Keluaran program dengan input aa
(Sumber: arsip penulis)

```
Masukkan kata sandi yang akan di brute force aaa

Mencoba kata sandi dengan panjang 1 karakter...
Waktu: 0.00 detik

Mencoba kata sandi dengan panjang 2 karakter...
Waktu: 0.00 detik

Mencoba kata sandi dengan panjang 3 karakter...
Waktu: 0.00 detik

Kata sandi: aaa
Waktu yang diperlukan: 0.02 detik
```

Gambar 8: Keluaran program dengan input aaa
(Sumber: arsip penulis)

```
Masukkan kata sandi yang akan di brute force aaaaa

Mencoba kata sandi dengan panjang 1 karakter...
Waktu: 0.00 detik

Mencoba kata sandi dengan panjang 2 karakter...
Waktu: 0.00 detik

Mencoba kata sandi dengan panjang 3 karakter...
Waktu: 0.00 detik

Mencoba kata sandi dengan panjang 4 karakter...
Waktu: 0.08 detik

Mencoba kata sandi dengan panjang 5 karakter...
Waktu: 8.25 detik

Kata sandi: aaaaa
Waktu yang diperlukan: 96.17 detik
```

Gambar 9: Keluaran program dengan input aaaaa
(Sumber: arsip penulis)

Tabel 2: Masukan dan waktu yang diperlukan

Nomor	Masukan	Waktu yang Diperlukan (detik)
1	a	0.00
2	aa	0.00
3	aaa	0.02
4	aaaa	1.26
5	aaaaa	96.17

Untuk masukan “123456” yang merupakan kata sandi paling popular, diperlukan 1758.11 detik. Sedangkan untuk masukan “password” tidak dapat dipecahkan dalam delapan jam pengujian.

```
Masukkan kata sandi yang akan di brute force 123456

Kata sandi: 123456
Waktu yang diperlukan: 1758.11 detik
```

Gambar 10: Keluaran program dengan input 123456
(Sumber: arsip penulis)

Dari hasil percobaan, kata sandi dengan enam karakter dapat dipecahkan menggunakan algoritma *brute force* murni dalam waktu sekitar 30 menit. Dengan algoritma penyerangan yang sebenarnya, kata sandi tersebut dapat dipecahkan degan jauh lebih cepat.

VI. KESIMPULAN DAN SARAN

Kata sandi yang kita gunakan sehari-hari harus lebih diperhatikan keamanannya. Keamanan kata sandi dapat ditingkatkan dengan menggunakan kata sandi yang lebih panjang dan menggunakan lebih banyak variasi jenis karakter, serta dengan menggunakan kata-kata yang jarang digunakan dan tida berhubungan dengan diri kita sendiri.

Dari sisi pemilik situs atau penyedia aplikasi, kata sandi dapat diamankan dengan menyimpannya dalam bentuk terenkripsi dengan fungsi *hashing* yang kuat dan menggunakan *salt*, serta dengan mengimplementasikan fitur-fitur anti-*brute force* seperti menonaktifkan akun setelah beberapa percobaan memasukkan kata sandi yang salah.

VI. LAMPIRAN

Tautan *source code* yang digunakan; https://github.com/florars/Makalah-Matdis/blob/main/testMatDis.py

REFERENCES

[1] Blog, Mail. com. (2022, November 28). Who invented passwords? A history of the security password. History of the password | mail.com blog. https://www.mail.com/blog/posts/history-of-security-password/146/

[2] Bosnjak, L., Sres, J., & Brumen, B. (2018). Brute-Force and dictionary attack on hashed real-world passwords. 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). https://doi.org/10.23919/mipro.2018.8400211

[3] Claburn, T. (2019, March 5). Use an 8-char windows NTLM password? don't. every single one can be cracked in under 2.5hrs. The Register® - Biting the hand that feeds IT. https://www.theregister.com/2019/02/14/password_length/

[4] Dell'Amico, Matteo, et al. "Password Strength: An Empirical Analysis." 2010 Proceedingss IEEE INFOCOM, 2010, doi:10.1109/infcom.2010.5461951.

[5] Dineshpathak, A. (2022, February 19). Algorithmic complexity. Devopedia. https://devopedia.org/algorithmic-complexity#Vilches-2015

[6] Fortinet. (2023). What is a brute force attack?: Definition, Types & How It Works. https://www.fortinet.com/resources/cyberglossary/brute-force-attack

[7] GeeksforGeeks. (2023, November 2). Analysis of algorithms: Big - Θ (big theta) notation. GeeksforGeeks. https://www.geeksforgeeks.org/analysis-of-algorithms-big-theta-notation/

[8] Jester, T. (2023, November 13). Understanding rockyou.txt: A tool for security and a weapon for hackers. Keeper Security Blog - Cybersecurity News & Product Updates. https://www.keepersecurity.com/blog/2023/08/04/understanding-rockyou-txt-a-tool-for-security-and-a-weapon-for-hackers/

[9] Kapersky. (2023, June 30). Brute Force attack: Definition and examples. www.kaspersky.com. https://www.kaspersky.com/resource-center/definitions/brute-force-attack

[10] Khan Academy. (n.d.). Big-Ω (big-omega) notation (article). Khan Academy. https://www.khanacademy.org/computing/computer-science/algorithms/asymptotic-notation/a/big-big-omega-notation

[11] Kristian. (n.d.). Bits of entropy - the importance of complex passwords. A Business Risk Approach To Information and Cyber Security. https://www.securitycentric.com.au/blog/bits-of-entropy-the-importance-of-complex-passwords

[12] Munir, Rinaldi."http://informatika.stei.itb.ac.id/~rinaldi.munir/"

[13] Passwords and authentication research. CyLab Usable Privacy and Security Lab (CUPS). (n.d.). http://cups.cs.cmu.edu/passwords.html

[14] A research paper - cyberhound. (n.d.). https://cyberhound.com/wp-content/uploads/CH-Research-Paper-Password-Security-LR-.pdf

[15] Top 200 most common passwords list. NordPass. (n.d.). https://nordpass.com/most-common-passwords-list/

[16] Vugdelija N., Nedeljković N., Kojić N., Luka Lukić L., Vesić M., "REVIEW OF BRUTE-FORCE ATTACK AND PROTECTION TECHNIQUES". unpublished.

[17] What is password entropy?. Proton. (2023, October 23). https://proton.me/blog/what-is-password-entropy

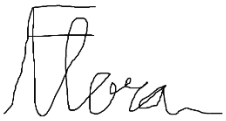
[18] What is password-based authentication?. Descope. (n.d.). https://www.descope.com/learn/post/password-authentication

[19] Zindros, D. (n.d.). A Gentle Introduction to Algorithm Complexity Analysis. A gentle introduction to algorithm complexity analysis. https://discrete.gr/complexity/

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 2023



Maria Flora Renata S 13522010