Pencarian Kata Sandi Menggunakan Pendekatan Brute Force: Studi Kasus dan Implikasi Keamanan

Muhammad Dicky Isra - 13523075
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: mdikiisra98@gmail.com , 13523075@std.stei.itb.ac.id

Abstrak*—Serangan *brute force* merupakan salah satu metode serangan siber paling fundamental, tetapi tetap menjadi ancaman yang persisten dan signifikan dalam lanskap keamanan digital modern. Makalah ini bertujuan untuk melakukan analisis terhadap pendekatan *brute force* dalam konteks pencarian kata sandi, mengkaji mulai dari landasan teoretis algoritmik hingga implementasi praktis dan implikasi keamanannya. Melalui tinjauan pustaka yang komprehensif, makalah ini menguraikan definisi, karakteristik, dan analisis kompleksitas dari algoritma *brute force* dan *exhaustive search*. Selanjutnya, metodologi serangan didemonstrasikan melalui contoh program Python untuk serangan *brute force* murni dan serangan kamus (*dictionary attack*), yang menyoroti evolusi taktik dari upaya komputasi mentah menjadi eksploitasi psikologi pengguna yang lebih cerdas. Analisis studi kasus nyata, termasuk insiden keamanan besar yang menimpa T-Mobile, Adobe, dan Dunkin' Donuts, digunakan untuk mengilustrasikan dampak destruktif dari serangan ini di dunia nyata, mulai dari pencurian data massal hingga kerugian finansial yang signifikan. Berdasarkan analisis tersebut, makalah ini merumuskan kerangka kerja mitigasi keamanan yang komprehensif dan berlapis, yang mencakup kebijakan kata sandi modern sesuai panduan NIST, kontrol teknis di tingkat server seperti *rate limiting* dan CAPTCHA, serta penekanan krusial pada implementasi Autentikasi Multi-Faktor (MFA) sebagai benteng pertahanan terkuat. Kesimpulan dari studi ini menegaskan bahwa meskipun secara teoretis sederhana, efektivitas praktis dari serangan *brute force* diperkuat oleh kelemahan manusia dalam memilih kata sandi, sehingga menuntut pendekatan pertahanan yang holistik dan tidak dapat ditawar lagi.

Kata Kunci—brute force, dictionary attack, keamanan kata sandi, kriptografi, exhaustive search, keamanan siber, autentikasi multi-faktor (MFA), mitigasi serangan

I. PENDAHULUAN

Dalam ekosistem digital yang semakin terintegrasi, proses autentikasi berfungsi sebagai gerbang utama yang mengontrol akses ke hampir setiap aset digital. Mulai dari akun email pribadi, data perbankan, hingga jaringan korporat dan infrastruktur kritis, mekanisme verifikasi identitas menjadi pilar fundamental dalam menjaga kerahasiaan, integritas, dan ketersediaan informasi. Di antara berbagai metode autentikasi, kombinasi nama pengguna dan kata sandi tetap menjadi yang paling umum digunakan. Namun, popularitasnya ini juga menjadikannya sebagai target utama serangan, di mana kata sandi sering kali menjadi tautan terlemah dalam rantai keamanan sebuah sistem. Keamanan seluruh sistem sering kali bergantung pada kekuatan satu kredensial ini, menjadikannya titik tunggal kegagalan (*single point of failure*) yang sangat rentan.

Di tengah lanskap ancaman siber yang terus berkembang dengan serangan-serangan canggih, serangan *brute force* tetap menjadi salah satu ancaman paling persisten dan mendasar. Algoritma *brute force* didefinisikan sebagai pendekatan yang lurus atau lempang (*straightforward*) untuk memecahkan suatu persoalan dengan cara yang sangat sederhana, langsung, dan jelas. Pendekatan ini bekerja dengan mencoba setiap kemungkinan solusi secara sistematis hingga solusi yang benar ditemukan. Dalam konteks keamanan, ini berarti mencoba setiap kemungkinan kombinasi kata sandi.[1]

Paradoks utama yang melingkupi serangan *brute force* adalah kontradiksi antara kesederhanaan teoretisnya dengan efektivitas praktisnya. Di satu sisi, algoritma ini sering dianggap "naif" dan tidak efisien karena membutuhkan sumber daya komputasi yang besar, terutama untuk ruang pencarian yang luas. Namun di sisi lain, studi kasus di dunia nyata secara konsisten menunjukkan bahwa serangan ini tetap sangat berhasil dan menjadi penyebab utama banyak insiden pelanggaran data berskala besar. Keberhasilan ini tidak hanya didorong oleh peningkatan eksponensial dalam kekuatan komputasi, tetapi juga oleh faktor manusia yang krusial: kecenderungan pengguna untuk memilih kata sandi yang lemah dan dapat diprediksi.

Berdasarkan latar belakang dan identifikasi masalah tersebut, makalah ini bertujuan untuk memberikan analisis yang komprehensif dan mendalam mengenai serangan *brute force* untuk pencarian kata sandi. Tujuan spesifik dari makalah ini adalah sebagai berikut:

1. Mengkaji landasan teoretis algoritma *brute force* dan *exhaustive search*, termasuk definisi, karakteristik, dan analisis kompleksitasnya.
2. Mendemonstrasikan implementasi praktis dari serangan *brute force* murni dan *dictionary attack* melalui contoh kode Python untuk memberikan pemahaman teknis tentang mekanisme serangan.
3. Menganalisis studi kasus nyata dari insiden pelanggaran data yang disebabkan oleh serangan *brute force* untuk mengilustrasikan dampak dan konsekuensi nyata dari serangan ini.
4. Merumuskan kerangka kerja mitigasi keamanan yang komprehensif dan berlapis, mencakup rekomendasi

kebijakan, kontrol teknis, dan strategi arsitektur pertahanan.

Ruang lingkup makalah ini mencakup pembahasan dari aspek teoretis-algoritmik hingga implementasi praktis dan strategi pertahanan siber, dengan fokus utama pada konteks pencarian kata sandi.

II. Landasan Teori

Untuk memahami secara mendalam bagaimana serangan pencarian kata sandi bekerja, penting untuk terlebih dahulu membangun fondasi teoretis yang kuat mengenai algoritma yang mendasarinya. Bagian ini akan mengkaji definisi, karakteristik, dan konsep-konsep kunci dari algoritma *brute force* dan *exhaustive search* berdasarkan materi referensi yang ada.

A. Definisi dan Karakteristik Algoritma Brute Force

Secara formal, algoritma *brute force* adalah pendekatan pemecahan masalah yang paling mendasar dan langsung. Ia didefinisikan sebagai metode yang lurus (*straightforward*) dan lempang, yang memecahkan masalah dengan cara yang sangat sederhana, langsung, dan jelas (*obvious way*). Filosofi di balik pendekatan ini dapat diringkas dengan slogan "Just do it!" atau "Just solve it!". Algoritma ini biasanya dapat dirumuskan secara langsung berdasarkan pernyataan masalah (*problem statement*) atau definisi dari konsep-konsep yang terlibat di dalamnya. Karena kesederhanaannya, algoritma ini sering kali menjadi titik awal dalam pengembangan solusi untuk suatu masalah.[1]

Karakteristik algoritma *brute force* dapat dianalisis dari dua sisi, yaitu kelebihan dan kelemahannya. Kelebihannya antara lain adalah sebagai berikut:

1. Penerapan Luas (*Wide Applicability*): Hampir semua persoalan dapat diselesaikan dengan menggunakan algoritma *brute force*. Keuniversalan ini menjadikannya metode yang sangat andal ketika tidak ada solusi yang lebih canggih yang diketahui.[1]
2. Kesederhanaan: Algoritma ini sederhana untuk dirancang dan mudah untuk diimplementasikan. Logikanya yang langsung membuatnya mudah dipahami dan di-debug.[1]
3. Menghasilkan Algoritma Baku: Untuk banyak tugas komputasi fundamental seperti pencarian, pengurutan (misalnya, *Selection Sort*, *Bubble Sort*), perkalian matriks, dan pencocokan string, algoritma *brute force* menyediakan solusi standar yang fungsional.[1]
4. Basis Perbandingan: Karena kesederhanaannya, algoritma *brute force* sering digunakan sebagai tolok ukur (*benchmark*) untuk membandingkan kinerja algoritma lain yang lebih efisien atau "cerdas". [1]

Kemudian, untuk kekurangannya adalah sebagai berikut:

1. Tidak Efisien: Algoritma *brute force* umumnya tidak "cerdas" dan tidak sangkil (*inefficient*). Ia sering kali membutuhkan biaya komputasi yang sangat besar, baik dari segi waktu eksekusi maupun penggunaan memori, terutama untuk masukan (*input*) berukuran besar.[1]
2. Tidak Praktis untuk Masalah Skala Besar: Kinerja yang lambat membuatnya tidak dapat diterima untuk banyak aplikasi dunia nyata yang menangani data dalam jumlah besar.[1]
3. Kurang Kreatif: Dibandingkan dengan strategi pemecahan masalah lain seperti *divide and conquer* atau pemrograman dinamis, *brute force* tidak menawarkan pendekatan yang konstruktif atau kreatif, melainkan hanya mengandalkan "tenaga" (*force*) komputasi. [1]

Sifat ini diringkas dengan baik oleh kutipan dari Ken Thompson, salah satu penemu sistem operasi UNIX: "When in doubt, use brute force". Ini menggarisbawahi perannya sebagai metode dasar yang dapat diandalkan, meskipun bukan yang paling elegan.[1]

B. Konsep Exhaustive Search (Pencarian Menyeluruh)

Exhaustive search, atau pencarian menyeluruh, adalah penerapan spesifik dari paradigma *brute force* untuk menyelesaikan persoalan-persoalan kombinatorik. Persoalan ini melibatkan pencarian solusi di antara objek-objek kombinatorik seperti permutasi, kombinasi, atau himpunan bagian dari sebuah himpunan. Pencarian kata sandi adalah contoh klasik dari masalah yang dapat diselesaikan dengan *exhaustive search*. [1], [2]

Metodologi *exhaustive search* dapat diuraikan menjadi tiga langkah fundamental [1]:

1. Enumerasi (Enumeration): Buat daftar (enumerasi) setiap kemungkinan solusi dengan cara yang sistematis. Dalam konteks kata sandi, ini berarti menghasilkan setiap kemungkinan kombinasi karakter.
2. Evaluasi (Evaluation): Evaluasi setiap kemungkinan solusi satu per satu. Untuk pencarian kata sandi, ini berarti mencoba setiap kombinasi yang dihasilkan untuk melihat apakah cocok dengan kata sandi target.
3. Seleksi (Selection): Setelah semua kemungkinan dievaluasi (atau ketika solusi ditemukan), umumkan solusi terbaik yang ditemukan. Dalam kasus pencarian kata sandi, proses berhenti saat kombinasi yang benar ditemukan.

Konsep *exhaustive search* memiliki penerapan langsung dan krusial dalam bidang kriptografi dan keamanan siber. Di sini, teknik ini digunakan oleh penyerang untuk menemukan kunci dekripsi atau kata sandi dengan cara mencoba semua kemungkinan kunci atau kata sandi yang ada. Serangan semacam ini dikenal dengan nama *exhaustive key search attack* atau, lebih umum, *brute force attack [1]*. Metode yang sama juga digunakan untuk menemukan PIN, kode akses, dan rahasia lainnya.

C. Analisis Kompleksitas: Mengukur Kelayakan Serangan

Kelayakan praktis dari sebuah serangan *brute force* ditentukan oleh analisis kompleksitas algoritmiknya. Kompleksitas waktu mengukur bagaimana waktu eksekusi sebuah algoritma tumbuh seiring dengan bertambahnya ukuran

masukan. Dalam konteks *brute force*, ukuran masukan ini sering kali terkait dengan panjang kata sandi dan ukuran set karakter yang digunakan.

Materi kuliah memberikan beberapa contoh untuk mengilustrasikan bagaimana kompleksitas dapat bervariasi secara dramatis [1]:

- Kompleksitas Polinomial: Masalah seperti pengurutan (*Selection Sort*) memiliki kompleksitas $O(n^2)$, dan perkalian matriks memiliki kompleksitas $O(n^3)$. Meskipun meningkat, pertumbuhan ini masih dapat dikelola untuk ukuran *n* yang wajar.
- Kompleksitas Eksponensial dan Faktorial: Masalah yang lebih sulit, seperti *Travelling Salesperson Problem* (TSP), memiliki kompleksitas faktorial O(n!), dan *0/1 Knapsack Problem* memiliki kompleksitas eksponensial $O(n \cdot 2^n)$. Pertumbuhan ini sangat cepat sehingga membuat solusi *brute force* menjadi tidak praktis bahkan untuk nilai *n* yang relatif kecil.

Untuk pencarian kata sandi, kompleksitasnya adalah $O(C^L)$, di mana *L* adalah panjang kata sandi dan *C* adalah jumlah karakter dalam set yang digunakan (misalnya, 26 untuk huruf kecil, 52 untuk huruf besar dan kecil, 62 untuk alfanumerik, dst.). Pertumbuhan eksponensial inilah yang secara teoretis membuat kata sandi yang panjang menjadi aman.

Contoh nyata yang kuat dari bagaimana kompleksitas dapat membuat serangan *brute force* menjadi tidak layak secara teoretis adalah kasus algoritma enkripsi DES. DES menggunakan kunci sepanjang 56 bit yang efektif. Ini berarti jumlah total kemungkinan kunci adalah 256, atau sekitar 72 kuadriliun (72.057.594.037.927.936) kombinasi [2]. Jika sebuah komputer dapat mencoba satu kunci setiap mikrodetik (satu juta kunci per detik), masih dibutuhkan lebih dari 2.284 tahun untuk mencoba semua kemungkinan. Jika hanya satu kunci per detik, waktu yang dibutuhkan adalah sekitar 2,28 miliar tahun. Contoh ini dengan jelas menunjukkan bagaimana ruang pencarian yang cukup besar dapat membuat serangan *brute force* murni menjadi tidak praktis.

Di sinilah letak paradoks sentral dari serangan *brute force*. Secara teoretis, seperti yang ditunjukkan oleh kasus DES, serangan ini seharusnya "tidak sangkil" (*inefisien*) dan mudah digagalkan dengan menggunakan kunci atau kata sandi yang cukup panjang. Namun, dalam praktiknya, serangan ini terus berhasil. Resolusi dari paradoks ini terletak pada tiga faktor dunia nyata yang mengubah kalkulasi teoretis [2]:

1. Peningkatan Kekuatan Komputasi: Hukum Moore dan kemajuan dalam komputasi paralel (menggunakan GPU) berarti bahwa jumlah percobaan yang dapat dilakukan per detik telah meningkat secara dramatis, membuat apa yang tidak layak kemarin menjadi layak hari ini. [3], [4]
2. Alat Serangan yang Dioptimalkan: Perangkat lunak seperti Hashcat dan John the Ripper adalah alat yang sangat dioptimalkan yang dapat melakukan miliaran atau bahkan triliunan tebakan per detik, tergantung pada perangkat keras dan algoritma *hashing* yang digunakan.[5]
3. Faktor Manusia: Ini adalah faktor yang paling krusial. Manusia secara inheren buruk dalam menciptakan keacakan. Kita memilih kata sandi yang mudah diingat, yang berarti kata sandi tersebut sering kali pendek, menggunakan kata-kata umum, atau mengikuti pola yang dapat diprediksi. Perilaku ini secara drastis mengurangi *ruang pencarian efektif*. Alih-alih harus mencari di seluruh ruang C^L, penyerang dapat fokus pada subset yang jauh lebih kecil dan memiliki probabilitas tinggi.[6]

Dengan demikian, dapat disimpulkan bahwa kerentanan utama bukanlah pada metode brute force itu sendiri, melainkan pada kegagalan manusia dan sistem untuk mempertahankan ruang pencarian yang cukup besar, acak, dan tidak dapat diprediksi untuk kredensial mereka. Serangan ini berhasil bukan karena algoritmanya cerdas, tetapi karena targetnya sering kali tidak aman.

III. Teknik Pencarian Kata Sandi

Setelah memahami landasan teoretisnya, bagian ini akan beralih ke aspek praktis dengan merinci metodologi teknis yang digunakan dalam serangan pencarian kata sandi. Akan dibahas dua pendekatan utama: serangan *brute force* murni yang mengandalkan kekuatan komputasi mentah, dan serangan kamus (*dictionary attack*) yang merupakan evolusi yang lebih cerdas dan efisien.

A. Serangan Brute Force Murni (Pure Brute Force Attack)

Serangan *brute force* murni adalah implementasi paling harfiah dari konsep *exhaustive search*. Mekanismenya adalah dengan secara sistematis menghasilkan dan menguji setiap kemungkinan kombinasi karakter dari sebuah set karakter yang telah ditentukan. Proses ini dapat dianalogikan dengan algoritma pencocokan string *brute force* yang dijelaskan dalam materi kuliah, di mana sebuah pola digeser satu karakter pada satu waktu dan dibandingkan dengan teks. [1]

Langkah-langkahnya adalah sebagai berikut:

1. Tentukan Set Karakter: Penyerang mendefinisikan set karakter yang akan digunakan, misalnya:
    - Hanya huruf kecil (abcdefghijklmnopqrstuvwxyz)
    - Huruf kecil dan besar
    - Alfanumerik (huruf kecil, besar, dan angka)
    - Semua karakter ASCII yang dapat dicetak (termasuk simbol seperti !@#$%^&*())
2. Tentukan Panjang Kata Sandi: Serangan biasanya dimulai dengan panjang minimum (misalnya, 1 karakter) dan berlanjut hingga panjang maksimum yang ditentukan.
3. Generasi dan Pengujian: Untuk setiap panjang, algoritma menghasilkan semua permutasi yang mungkin dari set karakter dan mengujinya satu per satu terhadap target (misalnya, hash kata sandi atau formulir login).

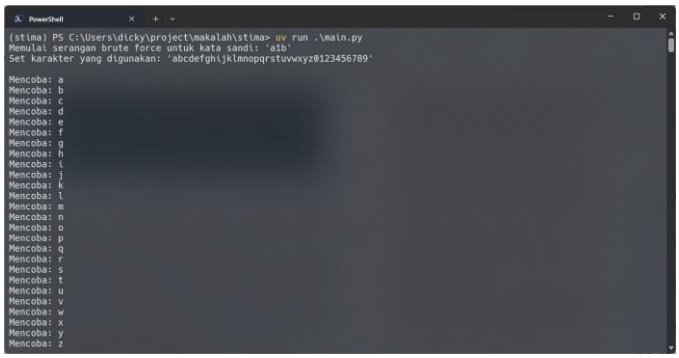
Kompleksitas serangan ini, seperti yang telah dibahas, adalah $O(C^L)$, yang membuatnya sangat tidak praktis untuk kata sandi yang bahkan cukup pendek.

B. Contoh Program Python (Brute Force Murni)

Berikut adalah contoh skrip Python sederhana yang mendemonstrasikan serangan *brute force* murni. Skrip ini bertujuan untuk menemukan kata sandi pendek untuk tujuan ilustrasi.

```python
def brute_force_attack(password_to_crack):
    chars = string.ascii_lowercase + string.digits
    print(f"Memulai serangan brute force untuk kata sandi: '{password_to_crack}'")
    print(f"Set karakter yang digunakan: '{chars}'\n")

    start_time = time.time()
    percobaan = 0

    for length in range(1, 9):
        for attempt in itertools.product(chars, repeat=length):
            attempt_str = "".join(attempt)
            print(f"Mencoba: {attempt_str}")
            percobaan += 1

            if attempt_str == password_to_crack:
                end_time = time.time()
                duration = end_time - start_time
                print("\n" + "=" * 40)
                print(f"Kata sandi berhasil ditemukan!")
                print(f"Kata Sandi: {attempt_str}")
                print(f"Waktu yang dibutuhkan: {duration:.4f} detik")
                print(f"Jumlah percobaan: {percobaan}")
                print("=" * 40)
                return attempt_str

    print("\nKata sandi tidak ditemukan dalam batas panjang yang ditentukan.")
    print(f"Jumlah percobaan: {percobaan}")
    return None


target_password = "a1b"
brute_force_attack(target_password)
```

Gambar 1. Kode program brute-force murni (sumber: dokumentasi pribadi)



```
PowerShell
(stima) PS C:\Users\dicky\project\makalah\stima> uv run .\main.py
Memulai serangan brute force untuk kata sandi: 'a1b'
Set karakter yang digunakan: 'abcdefghijklmnopqrstuvwxyz0123456789'

Mencoba: a
Mencoba: b
Mencoba: c
Mencoba: d
Mencoba: e
Mencoba: f
Mencoba: g
Mencoba: h
Mencoba: i
Mencoba: j
Mencoba: k
Mencoba: l
Mencoba: m
Mencoba: n
Mencoba: o
Mencoba: p
Mencoba: q
Mencoba: r
Mencoba: s
Mencoba: t
Mencoba: u
Mencoba: v
Mencoba: w
Mencoba: x
Mencoba: y
Mencoba: z
```

Gambar 2. Hasil kode program brute-force murni bagian atas (sumber: dokumentasi pribadi)

```
PowerShell
Mencoba: a0q
Mencoba: a0r
Mencoba: a0s
Mencoba: a0t
Mencoba: a0u
Mencoba: a0v
Mencoba: a0w
Mencoba: a0x
Mencoba: a0y
Mencoba: a0z
Mencoba: a00
Mencoba: a01
Mencoba: a02
Mencoba: a03
Mencoba: a04
Mencoba: a05
Mencoba: a06
Mencoba: a07
Mencoba: a08
Mencoba: a09
Mencoba: a1a
Mencoba: a1b

========================================
Kata sandi berhasil ditemukan!
Kata Sandi: a1b
Waktu yang dibutuhkan: 0.0677 detik
Jumlah percobaan: 2306
========================================
(stima) PS C:\Users\dicky\project\makalah\stima>
```

Gambar 3. Hasil kode program brute-force murni bagian bawah (sumber: dokumentasi pribadi)

Skrip di atas menggunakan pustaka itertools Python, khususnya fungsi product, yang sangat efisien untuk menghasilkan produk kartesius, dalam hal ini adalah semua kemungkinan kombinasi karakter. Meskipun efisien dalam generasi, kelemahan fundamental dari metode ini tetap terlihat. Menjalankan kode ini untuk kata sandi a1b akan selesai dalam hitungan milidetik. Namun, mencoba menemukan kata sandi 6 karakter alfanumerik (36 pilihan karakter) akan membutuhkan $366 \approx 2{,}17$ miliar percobaan. Ini dengan jelas menggarisbawahi mengapa serangan *brute force* murni tidak praktis untuk kata sandi dengan panjang dan kompleksitas yang wajar.

C. Serangan Kamus (Dictionary Attack)

Serangan kamus (*dictionary attack*) adalah evolusi yang jauh lebih cerdas dan efisien dari *brute force* murni. Alih-alih mencoba kombinasi acak, serangan ini menggunakan daftar kata (*wordlist*) yang telah disusun sebelumnya yang berisi tebakan-tebakan yang paling mungkin. Daftar ini, atau "kamus," secara drastis mengurangi ruang pencarian ke subset yang memiliki probabilitas keberhasilan tertinggi. [2], [7]

Sumber dari kamus ini bisa bermacam-macam:

- Daftar Kata Umum: Berisi kata-kata dari kamus bahasa (misalnya, Inggris, Indonesia).
- Kombinasi Informasi Pribadi: Informasi yang terkait dengan target, seperti nama, tanggal lahir, nama hewan peliharaan, alamat, dll.
- Kata Sandi yang Bocor: Ini adalah sumber yang paling kuat. Penyerang menggunakan daftar besar kata sandi yang telah bocor dari pelanggaran data sebelumnya. File rockyou.txt adalah contoh terkenal yang berisi lebih dari 14 juta kata sandi nyata yang bocor.
- Pola Umum: Kombinasi seperti "qwerty", "123456", atau "password123".

Serangan kamus pada dasarnya adalah eksploitasi terhadap psikologi manusia. Manusia tidak memilih kata sandi secara acak; kita memilihnya agar mudah diingat. Kecenderungan kognitif ini membuat kita rentan terhadap serangan yang menargetkan pola dan makna, bukan keacakan matematis. Keberhasilan serangan kamus adalah bukti langsung bahwa tautan terlemah dalam keamanan kata sandi adalah perilaku manusia itu sendiri.[8]

D. Contoh Program Python (Dictionary Attack)

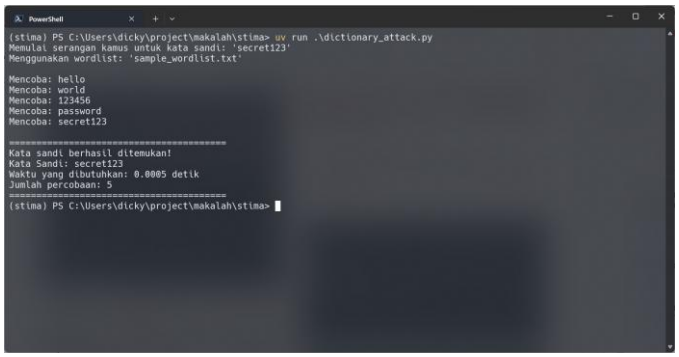
Berikut adalah contoh skrip Python yang mensimulasikan serangan kamus.

```python
def dictionary_attack(password_to_crack, wordlist_file):
    print(f"Memulai serangan kamus untuk kata sandi: '{password_to_crack}'")
    print(f"Menggunakan wordlist: '{wordlist_file}'\n")

    start_time = time.time()
    percobaan = 0

    try:
        with open(wordlist_file, "r", encoding="utf-8", errors="ignore") as f:
            for line in f:
                attempt = line.strip()
                print(f"Mencoba: {attempt}")
                percobaan += 1

                if attempt == password_to_crack:
                    end_time = time.time()
                    duration = end_time - start_time
                    print("\n" + "=" * 40)
                    print(f"Kata sandi berhasil ditemukan!")
                    print(f"Kata Sandi: {attempt}")
                    print(f"Waktu yang dibutuhkan: {duration:.4f} detik")
                    print(f"Jumlah percobaan: {percobaan}")
                    print("=" * 40)
                    return attempt

    except FileNotFoundError:
        print(f"Error: File wordlist '{wordlist_file}' tidak ditemukan.")
        return None

    print("\nKata sandi tidak ditemukan dalam wordlist.")
    print(f"Jumlah percobaan: {percobaan}")

    return None


target_password = "secret123"
wordlist_path = "sample_wordlist.txt"
dictionary_attack(target_password, wordlist_path)
```

Gambar 4. Kode program brute-force menggunakan kamus (sumber: dokumentasi pribadi)

```
(stima) PS C:\Users\dicky\project\makalah\stima> cat .\sample_wordlist.txt
hello
world
123456
password
secret123
admin
```

Gambar 5. Isi sample_wordlist.txt (sumber: dokumentasi pribadi)



```
PowerShell
(stima) PS C:\Users\dicky\project\makalah\stima> uv run .\dictionary_attack.py
Memulai serangan kamus untuk kata sandi: 'secret123'
Menggunakan wordlist: 'sample_wordlist.txt'

Mencoba: hello
Mencoba: world
Mencoba: 123456
Mencoba: password
Mencoba: secret123

========================================
Kata sandi berhasil ditemukan!
Kata Sandi: secret123
Waktu yang dibutuhkan: 0.0005 detik
Jumlah percobaan: 5
========================================
(stima) PS C:\Users\dicky\project\makalah\stima>
```

Gambar 6. Hasil kode program brute-force menggunakan kamus (sumber: dokumentasi pribadi)

Skrip ini membaca setiap baris dari file sample_wordlist.txt, menghapus *whitespace* atau karakter baris baru, dan membandingkannya dengan kata sandi target. Efisiensinya jauh melampaui *brute force* murni. Jika kata sandi target ada di dalam daftar, serangan ini akan berhasil dengan sangat cepat, terlepas dari panjang atau kompleksitas kata sandi tersebut. Keberhasilannya sepenuhnya bergantung pada kualitas dan kelengkapan kamus yang digunakan. Inilah sebabnya mengapa kebocoran data sangat berbahaya, karena setiap kata sandi yang bocor berpotensi menjadi amunisi untuk serangan kamus di masa depan.

E. Variasi Serangan Lanjutan

Penyerang modern sering kali tidak hanya mengandalkan satu metode, tetapi menggabungkan beberapa teknik untuk meningkatkan peluang keberhasilan.

- Hybrid Attacks: Serangan ini menggabungkan kekuatan serangan kamus dengan fleksibilitas *brute force*. Setelah mencoba kata-kata dari kamus, penyerang akan menerapkan aturan-aturan mutasi: menambahkan angka di akhir (misalnya, password -> password2024), mengganti huruf dengan simbol (misalnya, password -> p@ssw0rd), atau mengubah kapitalisasi. Taktik ini dirancang khusus untuk melewati kebijakan kompleksitas kata sandi yang naif.[9], [10]
- Credential Stuffing: Ini adalah salah satu serangan paling umum dan efektif saat ini. Penyerang mengambil daftar besar kredensial (pasangan nama pengguna dan kata sandi) yang dicuri dari satu pelanggaran data dan secara sistematis mencobanya di berbagai situs web dan layanan lain. Serangan ini mengeksploitasi kebiasaan buruk pengguna yang menggunakan kembali kata sandi yang sama di banyak akun. [8]
- Reverse Brute Force / Password Spraying: Berbeda dengan serangan tradisional yang mencoba banyak kata sandi untuk satu akun, *password spraying* membalik logika tersebut. Penyerang mengambil satu atau beberapa kata sandi yang sangat umum (misalnya, Password123, Spring2024!) dan mencobanya terhadap daftar nama pengguna. Taktik ini sangat efektif untuk menghindari kebijakan penguncian akun (*account lockout*) yang biasanya dipicu oleh terlalu banyak upaya gagal pada satu akun.[9], [11]

IV. STUDI KASUS: INSIDEN SERANGAN BRUTE FORCE DI DUNIA NYATA

Analisis teoretis dan demonstrasi teknis menjadi lebih bermakna ketika dihubungkan dengan insiden nyata. Bagian ini akan mengkaji beberapa studi kasus pelanggaran data terkenal yang disebabkan oleh berbagai bentuk serangan *brute force*. Kasus-kasus ini menyoroti dampak nyata dari serangan tersebut dan kerentanan yang dieksploitasi, mulai dari infrastruktur jaringan hingga akun pengguna akhir.

Tabel 1. Contoh Kasus Serangan

Target	Tahun	Tipe Serangan	Metode	Dampak & Konsekuensi
T-Mobile	2021	Brute Force	Serangan pada *login* SSH dari *gateway* GPRS yang tidak terlindungi.	Lebih dari 47 juta catatan pelanggan terekspos, termasuk informasi pribadi sensitif. Penyelesaian hukum senilai $31,4 juta. [9]
Adobe Systems	2013	Dictionary Attack	Eksploitasi kata sandi pengguna yang sangat lemah dan umum (misalnya, "123456").	Sekitar 153 juta catatan pengguna dicuri, menyebabkan kerusakan reputasi yang parah dan kerugian finansial.[7]
Dropbox	2012	Dictionary Attack / Credential Stuffing	Penggunaan kembali kata sandi yang bocor dari pelanggaran data di situs lain (LinkedIn).	Lebih dari 68 juta kredensial pengguna terekspos, menyoroti risiko sistemik dari penggunaan ulang kata sandi. [7]
Dunkin' Donuts	2015	Credential Stuffing	Penggunaan kredensial yang bocor dari situs lain untuk mengakses aplikasi loyalitas pelanggan.	Lebih dari 20.000 akun pelanggan disusupi, mengakibatkan pencurian dana loyalitas dan gugatan hukum dari negara bagian New York.[8], [9]

1) Kasus 1: Serangan Infrastruktur (T-Mobile, 2021)

Insiden T-Mobile pada Agustus 2021 adalah contoh nyata bagaimana serangan *brute force* dapat menargetkan lebih dari sekadar formulir login pengguna. Dalam kasus ini, penyerang, John Erin Binns, berhasil mendapatkan akses awal ke jaringan internal T-Mobile dengan melakukan serangan *brute force* terhadap *login* SSH (Secure Shell) pada sebuah *gateway* GPRS (General Packet Radio Service) di Washington yang tidak terlindungi dengan baik. [9]

Analisis dari kasus ini mengungkapkan sebuah poin kritis: permukaan serangan (*attack surface*) untuk serangan *brute force* jauh lebih luas daripada yang diperkirakan banyak orang. Sementara perhatian sering terfokus pada perlindungan akun pengguna akhir, antarmuka administratif, titik akses jaringan, dan layanan infrastruktur lainnya sering kali menjadi target yang lebih bernilai. Sistem-sistem ini mungkin tidak memiliki kontrol keamanan yang sama ketatnya, seperti Autentikasi Multi-Faktor (MFA) atau kebijakan penguncian akun yang agresif, yang diterapkan pada aplikasi yang menghadap pelanggan. Kompromi pada satu titik akses infrastruktur seperti ini dapat memberikan penyerang "kunci kerajaan," yang memungkinkan akses ke basis data masif yang berisi informasi pribadi sensitif dari puluhan juta pelanggan. Dampaknya sangat parah: lebih dari 47 juta catatan pelanggan saat ini, mantan pelanggan, dan calon pelanggan terekspos, termasuk nama, tanggal lahir, nomor Jaminan Sosial (SSN), dan detail SIM. Insiden ini berujung pada penyelesaian hukum senilai $31,4 juta. Kasus T-Mobile menjadi pengingat tegas bahwa strategi pertahanan harus bersifat holistik, mengeraskan setiap titik autentikasi di seluruh infrastruktur, bukan hanya yang paling jelas terlihat. [9]

2) Kasus 2: Eksploitasi Kata Sandi Lemah Skala Besar (Adobe, 2013 & Dropbox, 2012)

Insiden Adobe pada tahun 2013 adalah salah satu contoh paling ikonik dari serangan kamus (*dictionary attack*) yang berhasil dalam skala masif. Penyerang berhasil mencuri sekitar 153 juta catatan pengguna. Analisis pasca-insiden mengungkapkan bahwa keberhasilan serangan ini sangat difasilitasi oleh penggunaan kata sandi yang sangat lemah oleh pengguna. Kata sandi yang paling umum ditemukan dalam data yang bocor adalah "123456", diikuti oleh "123456789", "password", dan "adobe123". Serangan ini menunjukkan dengan jelas bahwa penyerang tidak perlu menggunakan teknik yang canggih ketika basis pengguna secara kolektif menciptakan kerentanan yang begitu besar melalui praktik penggunaan kata sandi yang buruk.[7]

Demikian pula, pelanggaran data Dropbox pada tahun 2012, yang mengakibatkan terungkapnya lebih dari 68 juta kredensial, berakar pada masalah yang sama. Penyerang menggunakan kata sandi yang sebelumnya telah bocor dari pelanggaran data di situs lain (terutama LinkedIn) dan melancarkan serangan kamus yang sangat tertarget terhadap akun Dropbox. Ini adalah contoh awal dari apa yang sekarang kita kenal sebagai *credential stuffing*. [7]

Kedua kasus ini secara kolektif menyoroti bagaimana efektivitas serangan kamus diperkuat oleh dua perilaku manusia yang saling terkait: pemilihan kata sandi yang lemah dan penggunaan kembali kata sandi di berbagai layanan. Dampaknya tidak hanya terbatas pada kerugian data, tetapi juga mencakup kerusakan reputasi yang signifikan dan kewajiban finansial yang besar bagi perusahaan yang terkena dampak.[7]

3) Kasus 3: Penggunaan Ulang Kata Sandi (Dunkin' Donuts, 2015)

Kasus Dunkin' Donuts pada tahun 2015 secara spesifik mengilustrasikan mekanisme dan dampak dari *credential stuffing*. Dalam insiden ini, Dunkin' Donuts sendiri tidak mengalami pelanggaran langsung pada basis datanya. Sebaliknya, penyerang menggunakan daftar kredensial (pasangan nama pengguna dan kata sandi) yang telah dicuri dari pelanggaran data di situs web lain dan secara otomatis mencobanya pada aplikasi loyalitas pelanggan Dunkin'. Karena banyak pelanggan menggunakan kembali kata sandi yang sama untuk akun Dunkin' mereka, serangan itu berhasil, mengakses lebih dari 20.000 akun. Penyerang kemudian dapat mencuri dana dari saldo kartu loyalitas pelanggan.[8], [9]

Insiden ini mengungkapkan sebuah kebenaran yang tidak nyaman dalam ekosistem digital yang saling terhubung: keamanan sebuah organisasi tidak sepenuhnya berada dalam kendalinya sendiri. Keamanannya juga bergantung pada praktik keamanan seluruh ekosistem digital dan kebersihan kata sandi para penggunanya. Pelanggaran di Situs A secara langsung menyediakan amunisi (kredensial yang valid) untuk serangan terhadap Situs B, C, dan D. Fenomena ini menciptakan risiko sistemik di mana "pelanggaran di mana saja adalah potensi pelanggaran di mana-mana." Hal ini menggarisbawahi kesia-siaan mengandalkan kata sandi saja sebagai mekanisme pertahanan. Ini membangun argumen terkuat untuk penerapan kontrol keamanan yang dapat memutus rantai risiko penggunaan ulang kata sandi, dengan Autentikasi Multi-Faktor (MFA) sebagai solusi utamanya. Akibat insiden ini, Dunkin' Donuts menghadapi gugatan hukum dari Jaksa Agung New York dan harus membayar denda serta meningkatkan protokol keamanannya secara signifikan. [8], [9]

V. Implikasi dan Mitigasi Keamanan

Analisis terhadap metodologi serangan dan studi kasus nyata menunjukkan bahwa serangan brute force, dalam berbagai bentuknya, memiliki implikasi yang serius dan luas. Memahami risiko ini adalah langkah pertama untuk membangun strategi pertahanan yang efektif. Bagian ini akan merinci dampak dan risiko serangan, diikuti dengan kerangka kerja mitigasi berlapis yang dirancang untuk mengatasi ancaman ini secara komprehensif.

A. Dampak dan Risiko Serangan

Konsekuensi dari serangan *brute force* yang berhasil dapat bersifat merusak bagi individu maupun organisasi. Berdasarkan temuan dari studi kasus dan riset keamanan lainnya, dampak utama dapat dikategorikan sebagai berikut:

- Pencurian Data (*Data Theft*): Ini adalah dampak yang paling langsung. Penyerang yang mendapatkan akses tidak sah dapat mencuri informasi sensitif, baik data pribadi pelanggan (nama, alamat, nomor Jaminan Sosial, informasi kartu kredit) maupun data korporat (rahasia dagang, kekayaan intelektual, data keuangan). [9], [12]
- Kerugian Finansial (*Financial Loss*): Kerugian finansial dapat terjadi melalui berbagai cara: pencurian dana secara langsung dari akun perbankan atau loyalitas , biaya pemulihan sistem, pembayaran denda peraturan (misalnya, GDPR, CCPA), biaya penyelesaian gugatan hukum , dan hilangnya pendapatan akibat gangguan bisnis.[8], [9]
- Penyebaran Malware: Akun yang berhasil diakses sering kali digunakan sebagai titik pijak (*foothold*) untuk melancarkan serangan lebih lanjut. Penyerang dapat menginstal *ransomware* untuk menyandera data, *spyware* untuk memata-matai aktivitas, atau menambahkan sistem yang terinfeksi ke dalam *botnet* untuk digunakan dalam serangan DDoS atau kampanye spam di masa depan.[12], [13]
- Kerusakan Reputasi (*Reputation Damage*): Pelanggaran data yang dipublikasikan dapat secara signifikan merusak reputasi sebuah organisasi. Hilangnya kepercayaan dari pelanggan, mitra, dan investor dapat memiliki dampak finansial jangka panjang yang bahkan lebih besar daripada biaya langsung dari insiden itu sendiri.[9], [12]

B. Strategi Pertahanan Berlapis (Layered Defense Strategy)

Tidak ada satu pun kontrol keamanan yang dapat memberikan perlindungan mutlak. Oleh karena itu, pendekatan pertahanan terbaik adalah strategi pertahanan mendalam (*defense-in-depth*), di mana beberapa lapisan kontrol keamanan bekerja sama untuk melindungi aset. Untuk melawan serangan *brute force*, strategi ini harus diterapkan di tiga tingkatan: kebijakan, server, dan arsitektur.

1) Tingkat Kebijakan: Membangun Fondasi Kata Sandi yang Kuat

Fondasi dari pertahanan terhadap serangan berbasis kata sandi adalah kebijakan yang kuat dan modern yang memandu pengguna dalam membuat dan mengelola kredensial mereka. Praktik terbaik saat ini sangat dipengaruhi oleh pedoman dari National Institute of Standards and Technology (NIST) dalam publikasi khususnya, SP 800-63B.[14]

- Prioritaskan Panjang di Atas Kompleksitas: Kebijakan lama yang memaksa pengguna untuk menyertakan kombinasi karakter huruf besar, huruf kecil, angka, dan simbol (*&(^%$)) kini dianggap kontraproduktif. Riset menunjukkan bahwa ketika dipaksa untuk memenuhi persyaratan kompleksitas, pengguna cenderung mengikuti pola yang dapat diprediksi, seperti mengkapitalisasi huruf pertama dan menambahkan angka atau simbol di akhir (misalnya, Password123!). Penyerang mengetahui pola ini dan memasukkannya ke dalam alat serangan hibrida mereka, yang secara efektif mengurangi ruang pencarian. Sebaliknya, NIST merekomendasikan untuk memprioritaskan panjang, dengan mendorong atau mewajibkan penggunaan kata sandi yang panjang (minimal 14-16 karakter) atau frasa sandi (*passphrase*). Frasa sandi yang panjang dan mudah diingat jauh lebih sulit untuk di-*brute force* daripada kata sandi pendek yang kompleks. [14]

- Larang Kata Sandi Umum: Sistem harus secara proaktif memblokir pengguna agar tidak memilih kata sandi yang terlalu umum atau yang telah diketahui bocor. Ini dapat dicapai dengan memeriksa kata sandi baru terhadap daftar kata sandi yang umum digunakan dan database kata sandi yang telah bocor, seperti yang disediakan oleh layanan seperti "Have I Been Pwned's Pwned Passwords".[15]
- Hindari Rotasi Kata Sandi Paksa: Praktik lama yang mengharuskan pengguna mengganti kata sandi mereka secara berkala (misalnya, setiap 90 hari) juga terbukti lebih berbahaya daripada bermanfaat. Kebijakan ini sering kali membuat pengguna hanya membuat modifikasi kecil dan dapat diprediksi pada kata sandi mereka yang ada (misalnya, Q12024!, Q22024!, Q32024!). NIST sekarang merekomendasikan untuk hanya mewajibkan perubahan kata sandi jika ada bukti kompromi.[16]

2) Tingkat Server: Kontrol Teknis untuk Menghambat Otomasi

Selain kebijakan yang baik, server harus dilengkapi dengan kontrol teknis yang dirancang untuk secara aktif mendeteksi dan menggagalkan upaya serangan otomatis.

- Pembatasan Laju (*Rate Limiting*): Ini adalah salah satu kontrol yang paling efektif. *Rate limiting* bekerja dengan membatasi jumlah upaya login yang dapat dilakukan dari satu alamat IP atau untuk satu akun dalam periode waktu tertentu (misalnya, tidak lebih dari 5 upaya gagal per menit). Dengan memperlambat laju tebakan secara drastis, kontrol ini membuat serangan *brute force* yang bergantung pada kecepatan menjadi tidak praktis dan memakan waktu sangat lama. [17]
- Kebijakan Penguncian Akun (*Account Lockout*): Setelah ambang batas upaya login yang gagal tercapai, akun dapat dikunci sementara. Namun, kebijakan ini harus diterapkan dengan hati-hati. Jika diterapkan secara naif, penyerang dapat menyalahgunakannya untuk melancarkan serangan *Denial of Service* (DoS) dengan sengaja mengunci akun-akun pengguna yang sah. Pendekatan yang lebih baik adalah menggunakan penundaan progresif (*progressive delays*), di mana waktu penguncian meningkat dengan setiap upaya gagal tambahan, atau memerlukan intervensi administrator untuk akun-akun kritis.[15], [17]
- Implementasi CAPTCHA: CAPTCHA (*Completely Automated Public Turing test to tell Computers and Humans Apart*) berfungsi sebagai penghalang penting untuk membedakan antara pengguna manusia dan bot otomatis. Dengan menyajikan tantangan yang mudah bagi manusia tetapi sulit bagi mesin (seperti mengidentifikasi teks terdistorsi atau memilih gambar tertentu), CAPTCHA dapat secara efektif menghentikan sebagian besar skrip *brute force*. Praktik terbaiknya adalah memicu tantangan CAPTCHA setelah beberapa upaya login yang gagal, sehingga menyeimbangkan keamanan dengan pengalaman pengguna.[18]

3) Tingkat Arsitektur: Autentikasi Multi-Faktor (MFA)

Lapisan pertahanan terkuat dan paling penting terhadap serangan berbasis kata sandi adalah Autentikasi Multi-Faktor (MFA). MFA harus dianggap sebagai persyaratan non-negosiasi untuk semua akun, terutama yang memiliki akses ke data sensitif.

- Mekanisme dan Efektivitas: MFA mengharuskan pengguna untuk memberikan dua atau lebih faktor verifikasi untuk membuktikan identitas mereka. Faktor-faktor ini biasanya termasuk:
    1. Sesuatu yang Anda Tahu: Kata sandi atau PIN.
    2. Sesuatu yang Anda Miliki: Kode dari aplikasi autentikator di ponsel, token fisik, atau kunci keamanan USB.
    3. Sesuatu yang Merupakan Bagian dari Diri Anda: Sidik jari, pemindaian wajah, atau biometrik lainnya. Dengan mewajibkan faktor kedua, MFA membuat kata sandi yang dicuri menjadi tidak berguna dengan sendirinya. Bahkan jika seorang penyerang berhasil melakukan *brute force* terhadap kata sandi, mereka tidak akan dapat mengakses akun tanpa memiliki akses fisik ke perangkat faktor kedua pengguna. Efektivitasnya sangat tinggi; sebuah laporan dari Microsoft menyatakan bahwa penerapan MFA dapat memblokir 99,9% serangan kompromi akun otomatis. [19], [20]
- MFA sebagai Pemutus Sirkuit untuk Risiko Sistemik: Seperti yang ditunjukkan oleh studi kasus *credential stuffing* , penggunaan ulang kata sandi menciptakan risiko sistemik di mana pelanggaran di satu tempat membahayakan banyak tempat lain. MFA berfungsi sebagai "pemutus sirkuit" (*circuit breaker*) yang efektif untuk risiko ini. Dengan membuat setiap proses login unik melalui faktor kedua yang dinamis, MFA secara efektif mengisolasi keamanan sebuah organisasi dari praktik kebersihan kata sandi yang buruk dari penggunanya dan dari kegagalan keamanan organisasi lain di internet. Ini bukan hanya perlindungan tingkat akun; ini adalah kontrol penting untuk mitigasi risiko sistemik di seluruh ekosistem digital.
- Evolusi Ancaman dan Pertahanan: Penting untuk diakui bahwa tidak ada pertahanan yang statis. Penyerang terus beradaptasi. Sebagai respons terhadap penyebaran MFA, taktik baru seperti serangan "kelelahan MFA" (*MFA fatigue*) atau "*push spam*" telah muncul. Dalam serangan ini, penyerang yang telah memiliki kata sandi akan berulang kali memicu permintaan persetujuan MFA ke ponsel pengguna, berharap pengguna yang lelah atau bingung pada akhirnya akan menyetujuinya. Hal ini menegaskan kembali pentingnya pendidikan pengguna yang berkelanjutan dan penerapan kebijakan MFA adaptif, yang dapat mengevaluasi konteks login (seperti lokasi, perangkat, dan waktu) untuk menerapkan kontrol yang lebih ketat pada permintaan berisiko tinggi.[20]

VI. Kesimpulan

Analisis yang telah dilakukan dalam makalah ini membawa pada serangkaian kesimpulan penting mengenai sifat, dampak, dan mitigasi serangan *brute force*. Meskipun berakar pada konsep algoritmik yang sederhana dan "lempang", serangan *brute force* dan variannya, terutama *dictionary attack*, tetap menjadi ancaman yang sangat nyata dan efektif dalam lanskap keamanan siber modern. Keberhasilan praktisnya yang berkelanjutan bukanlah cerminan dari kecanggihan metode itu sendiri, melainkan merupakan gejala dari dua faktor utama: kemajuan pesat dalam kekuatan komputasi dan, yang lebih signifikan, kelemahan persisten dalam perilaku manusia dalam membuat dan mengelola kata sandi.

Studi kasus T-Mobile, Adobe, Dropbox, dan Dunkin' Donuts secara kolektif mengilustrasikan spektrum penuh dari dampak serangan ini. Mulai dari kompromi infrastruktur kritis hingga eksploitasi massal akun pengguna melalui kata sandi yang lemah dan penggunaan ulang kredensial, konsekuensinya selalu parah, mencakup pencurian data berskala besar, kerugian finansial yang substansial, dan kerusakan reputasi jangka panjang. Temuan ini menegaskan bahwa mengandalkan satu lapisan pertahanan, terutama jika lapisan itu hanyalah kata sandi, adalah strategi yang ditakdirkan untuk gagal.

Oleh karena itu, kesimpulan utama dari makalah ini adalah keharusan mutlak untuk mengadopsi strategi pertahanan yang berlapis dan mendalam. Kerangka kerja pertahanan yang efektif harus mencakup:

1. Kebijakan Kata Sandi yang Cerdas: Beralih dari aturan kompleksitas yang usang ke kebijakan yang memprioritaskan panjang dan melarang kata sandi yang umum, sejalan dengan panduan modern dari NIST.
2. Kontrol Server yang Tangguh: Menerapkan mekanisme teknis seperti *rate limiting*, kebijakan penguncian akun yang bijaksana, dan CAPTCHA untuk secara aktif menghambat dan menggagalkan upaya serangan otomatis.
3. Implementasi Autentikasi Multi-Faktor (MFA) yang Menyeluruh: Ini adalah lapisan pertahanan yang paling krusial. MFA berfungsi sebagai pemutus sirkuit yang efektif terhadap serangan berbasis kata sandi, termasuk *credential stuffing*, dan secara drastis mengurangi permukaan serangan. Implementasinya tidak boleh lagi dianggap sebagai pilihan, melainkan sebagai standar keamanan dasar.

Sebagai pandangan ke depan, industri keamanan terus bergerak menuju solusi yang dapat mengurangi ketergantungan pada kata sandi yang dapat diingat manusia. Munculnya teknologi autentikasi tanpa kata sandi (*passwordless authentication*), seperti kunci keamanan FIDO2, biometrik, dan metode login berbasis perangkat, menjanjikan masa depan di mana seluruh kelas kerentanan yang dibahas dalam makalah ini dapat dimitigasi secara fundamental. Namun, hingga transisi tersebut selesai, prinsip-prinsip pertahanan berlapis yang diuraikan di sini akan tetap menjadi landasan penting untuk menjaga keamanan aset digital dari ancaman *brute force*.

Video Link di Youtube

https://youtu.be/gRhJhdrJXdU

Source Code di Github

https://github.com/DaDecky/Makalah-Stima

Ucapan Terima Kasih

Puji dan Syukur atas kehadirat Allah SWT yang telah banyak membantu penulis untuk dapat menyelesaikan makalah berjudul "Pencarian Kata Sandi Menggunakan Pendekatan Brute Force: Studi Kasus dan Implikasi Keamanan". Tak lupa penulis ucapkan terima kasih dan rasa syukur kepada :

1. Orang tua penulis yang selalu memberikan dukungan serta doa kepada penulis dalam kondisi apapun
2. Keluarga penulis yang senantiasa memberikan dukungan maupun doa.
3. Bapak Dr. Ir. Rinaldi Munir, M.T. selaku dosen mata kuliah Strategi Algoritma yang senantiasa membantu dan membimbing mahasiswa Strategi Algoritma Kelas K-02, dan juga telah membuat website yang sangat membantu dalam pengerjaan makalah ini.

Referensi

[1] R. Munir, "Algoritma Brute Force (Bagian 1)." Diakses: 23 Juni 2025. [Daring]. Tersedia pada: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/02-Algoritma-Brute-Force-(2025)-Bag1.pdf

[2] R. Munir, "Algoritma Brute Force (Bagian 2)." Diakses: 23 Juni 2025. [Daring]. Tersedia pada: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/03-Algoritma-Brute-Force-(2025)-Bag2.pdf

[3] D. Ambolis, "What Is Moore's Law, And How Does It Impact Cryptography?" Diakses: 24 Juni 2025. [Daring]. Tersedia pada: https://blockchainmagazine.net/what-is-moores-law-and-how-does-it-impact-cryptography/

[4] D. Goodin, "25-GPU cluster cracks every standard Windows password in <6 hours - Ars Technica." Diakses: 24 Juni 2025. [Daring]. Tersedia pada: https://arstechnica.com/information-technology/2012/12/25-gpu-cluster-cracks-every-standard-windows-password-in-6-hours/

[5] "Hashcat Password Cracking & Password Policy | Part 1 | ProSec GmbH." Diakses: 24 Juni 2025. [Daring]. Tersedia pada: https://www.prosec-networks.com/en/blog/hashcat-password-cracking-password-policy/

[6] "The Psychology of Password Generation - Schneier on Security." Diakses: 24 Juni 2025. [Daring]. Tersedia pada: https://www.schneier.com/blog/archives/2006/03/the_psychology.html

[7] "Real-World Examples of Dictionary Attacks and Their Impacts | SubRosa." Diakses: 23 Juni 2025.

[Daring]. Tersedia pada: https://www.subrosacyber.com/en/blog/real-world-dictionary-attack-examples
[8] J. Martinez, “What is a Brute Force Attack? Types, Examples & Prevention | StrongDM.” Diakses: 23 Juni 2025. [Daring]. Tersedia pada: https://www.strongdm.com/blog/brute-force-attack
[9] B. Buckman, “What is a Brute Force Attack? A Guide for IT Security Professionals | Huntress.” Diakses: 23 Juni 2025. [Daring]. Tersedia pada: https://www.huntress.com/cybersecurity-education/cybersecurity-101/topic/what-is-a-brute-force-attack
[10] “Dictionary Attack | BlackFog.” Diakses: 23 Juni 2025. [Daring]. Tersedia pada: https://www.blackfog.com/cybersecurity-101/dictionary-attack/
[11] “What Is a Brute Force Attack? | F5.” Diakses: 24 Juni 2025. [Daring]. Tersedia pada: https://www.f5.com/glossary/brute-force-attack
[12] “What Are Brute-Force Attacks? | Examples & Prevention Tips.” Diakses: 24 Juni 2025. [Daring]. Tersedia pada: https://sosafe-awareness.com/glossary/brute-force-attack/#How-to-prevent-brute-force-attacks
[13] B. Cleary, “What is a brute force attack? - Norton.” Diakses: 24 Juni 2025. [Daring]. Tersedia pada: https://us.norton.com/blog/emerging-threats/brute-force-attack
[14] V. Vicente, “NIST Password Guidelines.” Diakses: 24 Juni 2025. [Daring]. Tersedia pada: https://auditboard.com/blog/nist-password-guidelines/
[15] “Authentication - OWASP Cheat Sheet Series.” Diakses: 24 Juni 2025. [Daring]. Tersedia pada: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html
[16] “Password Policy Best Practices for Strong Security in AD.” Diakses: 24 Juni 2025. [Daring]. Tersedia pada: https://www.netwrix.com/password-policy-best-practices.html
[17] K. Tripathi, “Brute Force Attack: Preventing Trial-and-Error Logins - Seceon Inc.” Diakses: 24 Juni 2025. [Daring]. Tersedia pada: https://seceon.com/brute-force-attack-preventing-trial-and-error-logins/
[18] “What is Brute Force Attack?” Diakses: 24 Juni 2025. [Daring]. Tersedia pada: https://friendlycaptcha.com/wiki/what-is-brute-force-attack/
[19] M. Ibrahim, “The Multifaceted Benefits of Multi-Factor Authentication.” Diakses: 24 Juni 2025. [Daring]. Tersedia pada: https://supertokens.com/blog/benefits-of-multi-factor-authentication
[20] “What is Multifactor Authentication (MFA)? Attacks & Exploits.” Diakses: 24 Juni 2025. [Daring]. Tersedia pada: https://www.vaadata.com/blog/multifactor-authentication-mfa-how-does-it-work-types-of-attacks-exploits-and-security-best-practices/

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, 24 Juni 2025

Muhammad Dicky Isra dan 13523075