Optimized Dictionary Attack on MD5 *Algortihm*

Sandromedo Christa Nugroho
Badan Koordinasi Penanaman Modal
Jl. Jend. Gatot Subroto No. 44, Senayan, Kebayoran Baru, Jakarta, 12190, Indonesia
major.ruft@gmail.com

Abstract—***Applications and services could recognize and grant access for someone, based on the input of username and password given by authorized user. Therefore, in the implementation of information system both parameters are should be kept secret and protected from unauthorized parties. Before being stored in the database, generally passwords are secured using cryptographic techniques, that is by hashing username and password using hash function algorithm. Practically, there are still any possible attacks that could break hashed password stored in the database, one of the attack is dictionary attack. This paper would discuss about real dictionary attack and its optimization techniques on MD5 algorithm-hashing password, beside that it would also discuss about the need of time and memory in doing the attack.***

Keywords : Password Security; Hash Function Algorithm; Dictionary Attack and Its Optimization Techniques

Abstrak—***Secara sederhana sebuah aplikasi dan/atau layanan dapat mengenali dan memberikan akses pengguna berdasarkan pada input username dan password yang diberikan olehnya. Oleh karena itu, dalam implementasi sebuah sistem informasi kedua parameter tersebut merupakan hal yang harus dirahasiakan dan dilindungi dari pihak yang tidak berwenang. Sebelum disimpan kedalam database, umumnya password diamankan menggunakan teknik kriptografi, yaitu dengan melakukan hashing pada username dan password menggunakan algoritma fungsi hash. Namun pengamanan tersebut bukan merupakan pengamanan absolute, karena masih terdapat kemungkinan serangan yang dapat memecahkan hashing password yang tersimpan didalam database, salah satunya adalah dengan menerapkan dictionary attack. Pada tulisan ini akan dilakukan dictionary attack dan teknik optimalisasinya pada password yang telah dihash dengan menggunakan algoritma MD5, selain itu akan dibahas juga mengenai kebutuhan waktu dan memory dalam melakukan serangan tersebut.***

Kata Kunci : Keamanan* Password*, Algoritma Fungsi Hash, Dictionary Attack dan Teknik Optimalisasinya

I. INTRODUCTION

Secara umum paradigma pengguna terhadap sistem informasi yang telah diamankan dengan menggunakan teknik kriptografi adalah keamanan *absolute* yang tidak dapat dipecahkan dengan mudah dalam hitungan waktu terhitung. Namun sayangnya hal tersebut tidak berlaku apabila pihak penyerang menggunakan perangkat keras super cangih yang memiliki kemampuan komputasi sangat baik, seperti super komputer, komputer kuantum, paralel GPU NVIDIA, paralel FPGA, komputer kluster dan perangkat keras lainnya, karena dengan menggunakan perangkat-perangkat keras tersebut, teknik kriptografi apapun dapat dipecahkan dengan mudah, bahkan dalam waktu yang sangat cepat.

Serangan praktis yang paling simple dan sederhana untuk memecahkan teknik kriptografi adalah dengan menggunakan *brute force attack*, namun seperti yang kita ketahui, serangan tersebut memerlukan perangkat, waktu, data, daya (*power*) dan *source* yang sangat besar, karena pihak penyerang harus mencoba seluruh kemungkinan jawaban yang ada. Secara *best case* terdapat faktor keberuntungan dalam *brute force attack*, yaitu dengan hanya melakukan 1 kali percobaan, meskipun persentase faktor keberuntungan tersebut sangatlah kecil, yaitu hanya sebesar 1/2128 (untuk algoritma MD5). Terdapat serangan praktis lain yang memiliki kemungkinan berhasil lebih tinggi dan membutuhkan waktu yang relatif lebih cepat dibandingkan *brute force attack*, yaitu *dictionary attack*. *Dictionary attack* merupakan serangan praktis dengan menggunakan input terprediksi yang telah disimpan pada *hardisk* dan/atau *server storage*, sehingga dapat menghemat biaya komputasi, namun membutuhkan *source* media penyimpanan yang sangat besar.

Dictionary attack memiliki probabilitas keberhasilan serangan yang cukup tinggi, karena serangan tersebut melibatkan psikologi pengguna dalam melakukan penebakan *password*, dimana kebanyakan pengguna menggunakan kata-kata yang lazim terdapat dalam percakapan sehari-hari atau yang mereka ingat atau yang sering mereka ucapkan, serta tanggal-tanggal penting tertentu (semisal tanggal ulang tahun, tanggal pernikahan dan lain sebagainya). Selain itu *dictionary attack* juga dapat lebih dioptimalkan dengan menggunakan teknik pengkelasan, sehingga jumlah percobaan yang harus dilakukan oleh pihak penyerang dalam memecahkan teknik kriptografi dapat diperkecil.

II. LANDASAN TEORI

A. *Password*

Password adalah kumpulan karakter atau *string* yang digunakan oleh pengguna untuk memverifikasi/mengotentikasi identitasnya dalam menggunakan aplikasi atau layanan tertentu, dimana untuk menjaga kerahasiaan dan kemungkinan penyalahgunaan oleh pihak lain, maka *password* harus diganti secara periodik. Definisi lain dari *password* adalah kata kunci yang dirahasiakan nilainya, tujuannya adalah untuk mengamankan data atau akses pengguna terhadap program dari orang-orang yang tidak berhak [5]. *Password* adalah metode otentikasi yang mudah dan mudah bagi pengguna yang sedang masuk sistem komputer. Sistem ini hanya mengharuskan pengguna untuk menyajikan sesuatu yang dia ketahui sebagai abukti bahwa dia sebenarnya adalah orang yang dia klaim. Ini mudah diimplementasikan, tapi sama saja Waktu pendekatan *password* tunduk pada sejumlah ancaman

keamanan. Berikut ini adalah risiko keamanan umum dimana pengguna yang sah dapat kehilangan kata sandinya: Selain itu *password* juga dapat definisikan sebagai metode bagi pengguna untuk mengotentikasikan dirinya saat masuk sistem komputer atau situs web [8]. Secara umum *password* merupakan garis pertahanan pertama untuk mengcounter akses dari pengguna yang tidak berkepentingan, oleh karena itu keamanan *password* harus senantiasa dijaga dengan baik, salah satu cara untuk menjaga keamanan *password* adalah dengan menerapkan kebijakan manajemen *password* yang baik, antara lain :

1. Mengganti *username* dan *password* default yang dibangkitkan oleh sistem secara otomatis pada saat login pertama kali kedalam aplikasi, menggunakan *password* yang terdiri dari kombinasi huruf besar, huruf kecil, angka, tanda baca dan karakter khusus, contohnya "P4$$w0rD*!" untuk mempersulit *brute force attack*;
2. Menghindari penggunaan *password* yang berisikan kata-kata umum atau yang terdapat dalam kamus untuk mempersulit *dictionary attack*, menghindari penggunaan *password* yang mengandung karakteristik atau pengenal pribadi seperti nama panggilan, tanggal lahir, nama keluarga dan lain-lain. Namun sebaiknya *password* memiliki sifat mudah diingat, contohnya 13uD!m4N yang mudah diingat dengan menggunakan metode *mnemonic* dari nama Budiman;
3. Menggunakan *password* yang berbeda atau unik pada setiap aplikasi dan/atau layanan jaringan komunikasi, melakukan penyimpanan *password* pengguna pada aplikasi dan/atau tempat yang aman, contohnya aplikasi *password* safe, keyPass, *password* manager dan aplikasi lainnya, sehingga tidak perlu diingat, tidak mudah dicuri dan tidak dapat disalahgunakan oleh pihak yang tidak berkepentingan;
4. Menyusun *password* dengan minimal terdiri dari 8 karakter/lebih, untuk meningkatkan level keamanan terhadap *brute force attack* dan yang terakhir melakukan perubahan *password* secara periodik, untuk mempersulit penyerang melakukan pendugaan terhadap *password*.

B. Algoritma MD5

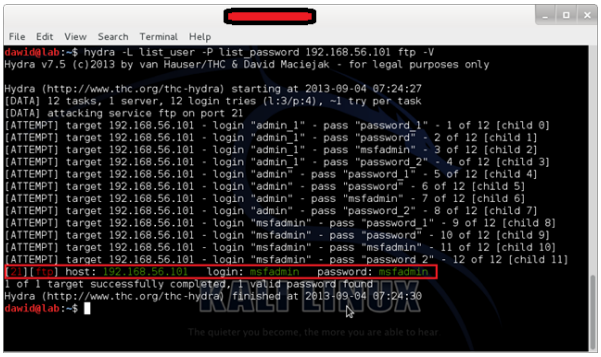
Algoritma MD5 adalah algoritma fungsi *hash* yang dikembangkan oleh Ronald L. Rivest pada tahun 1992. Algoritma MD5 dapat melakukan *hashing* pada pesan dengan panjang sembarang (*arbitrary*) menjadi *message digest* dengan panjang tetap (*fix*) sepanjang 128 bit. Algoritma tersebut memiliki lisensi RSA dan banyak digunakan oleh masyarakat umum secara luas pada berbagai aplikasi/layanan kriptografi berdasarkan pada dokumen RFC 1321. Pada tahun 1996 Hans Dobbertin menemukan beberapa kelemahan secara matematis pada algoritma MD5, kemudian pada tahun 2004 Xiaoyun Wang and Hongbo Yu berhasil membuktikan bahwa algoritma MD5 rentan terhadap serangan kolisi dan pada tahun yang sama J. Black, M. Cochran dan T. Highland menemukan cara untuk menemukan pasangan pesan berbeda yang dapat menghasilkan nilai *message digest* yang sama (*preimage attack*), bahkan pada tahun 2008 para ahli kriptografi dapat menemukan teknik untuk memalsukan sertifikat SSL yang menggunakan algoritma MD5 sebagai algoritma fungsi *hash*nya. Berdasarkan pada beberapa *security holes* tersebut, maka dokumen RFC 6331 tahun 2011 menyatakan bahwa algoritma MD5 tidak direkomendasikan untuk digunakan, sehingga masyarakat luas mulai beralih untuk menggunakan algoritma fungsi *hash* lainnya, seperti algoritma SHA-2. Namun untuk sektor pembelajaran dan uji coba teknik kriptografi, algoritma MD5 masih dapat digunakan dan dikaji lebih jauh, karena didukung oleh banyak *library* pada beberapa bahasa pemrograman, salah satunya adalah dengan melakukan *dictionary attack* pada algoritma tersebut. Tabel 1. dibawah menunjukkan spesifikasi algoritma fungsi *hash* MD5.

Tabel 1. Spesifikasi Algoritma Fungsi *Hash* MD5.

No.	Keterangan	Deskripsi
1.	Pengembang/Pempublikasi	Ronald Rivest
2.	Tahun Publikasi	1992
3.	Ukuran Pesan Input	*Arbitrary*
4.	Ukuran *Message Digest*	128 Bit
5.	*Round*	4
6.	Struktur	*Merkle-Damgard*
7.	Dokumen Standar	RFC 1321

C. Dictionary attack

Menurut Jeff Atwood, *dictionary attack* adalah teknik untuk memecahkan algoritma enkripsi atau mekanisme otentikasi dengan cara menentukan kunci dekripsi atau *frase* khusus dengan mencari kombinasi kata-kata yang paling memungkinkan yang terdapat pada sebuah kamus. Definisi lain dari *dictionary attack* menurut Imam Prabowo adalah teknik percobaan pemecahan kode dengan mencoba kombinasi kata yang paling mungkin berhasil, dengan input sebuah *list of words* yang dapat didefinisikan (disebut juga kamus) biasanya berasal dari daftar kombinasi kata-kata umum yang terdapat dalam kamus, misalnya kamus bahasa Inggris atau bahasa Indonesia. Selain itu *dictionary attack* juga didefinisikan oleh *Search Security* sebagai teknik/metode untuk memecahkan *password* pada komputer dan/atau *server* dengan menggunakan setiap kata pada kamus sebagai input *password*nya secara sistematis. Penulis sendiri mendefinisikan *dictionary attack* sebagai serangan praktis dengan menggunakan input terprediksi yang telah disimpan pada *hardisk* dan/atau *server storage*, sehingga dapat menghemat biaya komputasi, meskipun membutuhkan *source* media penyimpanan yang sangat besar. Terdapat beberapa *website* yang menyediakan layanan *dictionary attack* secara *online*, gambar 1. dibawah menunjukkan Hydra, salah satu *website dictionary attack online* yang cukup familiar dan banyak digunakan.



Gambar 1. Penggunaan Hydra
(Sumber : http://resources.infosecinstitute.com/online-dictionary-attack-with-hydra/#gref).

Implementasi *dictionary attack* memungkinkan penyerang untuk menghemat tenaga dan waktu dengan melakukan komputasi berdasarkan pada *list of* words atau disebut juga dengan *list of hash*es dalam serangan pada *password-password* pengguna.

III. METODE SERANGAN

Penerapan *dictionary attack* dalam melakukan penyerangan terhadap *password* pengguna merupakan langkah yang lebih efektif dan efisien dibandingkan dengan penerapan *dictionary attack*, karena perangkat serangan hanya perlu melakukan perbandingan antara nilai *message digest* yang ingin dipecahkan (dapat juga dicari *preimage*nya) dengan kumpulan *message digest* yang telah disimpan didalam media penyimpanan, dalam hal ini perangkat keras yang digunakan untuk menyerang memiliki kinerja yang lebih ringan, karena tidak perlu lagi menjalankan proses *hashing* terhadap pesan. Teknik dictionary dttack yang efektif dan efisien tersebut, masih dapat dioptimalkan kembali dengan menerapkan teknik pengkelasan terhadap kumpulan *message digest*.

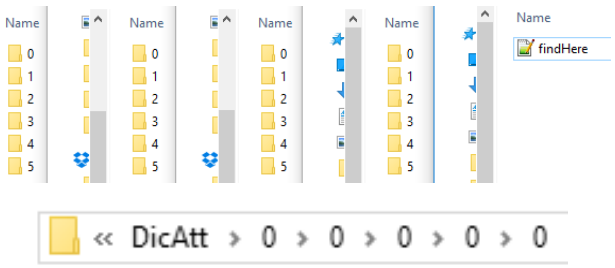
Pengkelasan adalah metode pengelompokkan kumpulan *message digest* berdasarkan pada nilai bytenya, seperti yang kita ketahui range nilai integer sebuah byte adalah nilai 0 sampai dengan nilai 255, berdasarkan hal tersebut maka 8 bit pertama pada *message digest* dapat dikelaskan kedalam 256 subkelas yang masing-masing subkelasnya berisi 8 bit kedua yang juga dapat dikelaskan kedalam 256 subkelas yang masing-masing subkelasya berisi 8 bit ketiga dan seterusnya sampai pada 8 bit terakhir. Tujuan utama dari teknik pengekelasan adalah mempersingkat dan mempercepat proses pencarian *message digest*, karena dilakukan secara terfokus pada hasil yang ingin dicapai. Sebagai contoh *message digest* dengan nilai “12 34 56 78 ... 90” cukup dicari pada kelas “12” lalu “34” kemudian “56” dan seterusnya sampai dengan maksimal kelas yang dapat dibentuk, sehingga menghindari pencarian yang sia-sia dan menjadi sebuah teknik optimalisasi pada *dictionary attack*. Adapun jumlah kelas dalam serang *dictionary attack* teknik pengkelasan pada algoritma fungsi *hash* berbeda-beda, tergantung pada panjang *message digest*nya. Jumlah kelas dapat di hitung dengan menggunakan persamaan :

class_count = *length* (*message digest*) / 8

Keterangan :

1. class_count = Banyaknya kelas yang diperlukan
2. *message digest* = Output algoritma fungsi *hash*
3. lenght *message digest* = Panjang dari output algoritma fungsi *hash* (umumnya dinyatakan dalam bit/byte)

Sebagai contoh panjang *message digest* algoritma MD5 adalah 128 bit, maka dalam serangan *dictionary attack* dibutuhkan kelas sebanyak 16 dengan masing-masing subkelas sebanyak 256, sedangkan untuk algoritma SHA-256 yang memiliki panjang *message digest* 256 bit, maka dibutuhkan kelas sebanyak 32 dengan masing-masing subkelas sebanyak 256. *Dictionary attack* teknik pengkelasan dapat lebih dioptimalkan dengan cara membentuk kelas dan subkelas berdasarkan pada nibble (4 bit), sehingga proses pencarian akan lebih singkat dan cepat, karena pendefinisian message diggest yang akan dicari *preimage*nya lebih detail. Namun teknik tersebut akan membutuhkan *memory* dan biaya yang lebih besar, karena harus membagi kelas 2 kali lebih besar dibandingkan dengan pembentukan kelas dan subkelas berdasarkan pada byte (8 bit). Gambar 2. dibawah menunjukkan contoh pembagian kelas dan subkelas pada *dictionary attack* teknik pengkelasan.



Gambar 2. Pembagian Kelas dan Subkelas pada *Dictionary Attack* Teknik Pengkelasan.

Tabel 2. dibawah menunjukkan algoritma fungsi *hash*, jumlah kelas dan subkelas yang dibutuhkan untuk melakukan *dictionary attack* teknik pengkelasan.

Tabel 2. Kebutuhan Dalam *Dictionary Attack* Teknik Pengkelasan.

No.	Algoritma Fungsi *Hash*	Ukuran Block Output	Jumlah Kelas	Jumlah Subkelas
1.	MD5	128 Bit	16	4.096
2.	RIPEMD 128	128 Bit	16	4.096
3.	RIPEMD 160	160 Bit	20	5.120
4.	RIPEMD 256	256 Bit	32	8.192
5.	RIPEMD 320	320 Bit	40	10.240
6.	SHA-1	160 Bit	20	5.120
7.	SHA 224	224 Bit	28	7.168
8.	SHA 256	256 Bit	32	8.192
9.	SHA 384	384 Bit	48	12.288
10.	SHA 512	512 Bit	64	16.384
11.	Tiger	192 Bit	24	6.144

IV. PARAMETER SERANGAN

Parameter *dictionary attack* terhadap algoritma MD5 akan dilihat berdasarkan pada 4 hal, antara lain : spesifikasi perangkat dan kebutuhan penyimpanan (*memory*) serangan.

A. *Spesifikasi Perangkat*

Spesifikasi perangkat merupakan penjabaran mengenai spesifikasi perangkat lunak dan keras yang digunakan dalam melakukan *dictionary attack* pada algoritma MD5. Tabel 3. dibawah menunjukkan spesifikasi perangkat keras.

Tabel 3. Spesifikasi Perangkat Keras.

No.	Informasi Sistem	Keterangan
1.	Manufaktur Sistem	ASUSTeK Computer Inc.
2.	Model	N552VX
3.	BIOS	N552VX.205
4.	Processor	Intel(R) Core(TM) i7-6700HQ CPU @ 2.66GHz (8 CPUs)
5.	Memory RAM	8192 MB
6.	DirectX Version	DirectX 12

Sedangkan tabel 4. dibawah menunjukkan kondisi dan setting perangkat lunak.

Tabel 4. Kondisi Dan Setting Perangkat Lunak.

No.	Informasi Sistem	Keterangan
1.	Sistem Operasi	*Windows 10 Home 64-bit* (10.0, *Build* 16299)
2.	*Maximum Processor State*	100 %
3.	*Minimum Processor State*	100 %
4.	*Battery*	*Plugged In*
5.	*Misvellaneous Power Setting*	*Never Sleep* *Never Hibernate* *20 Minutes Turn Off HDD* *Never Turn Off Display*
6.	*Brightness*	10 (*Maximum*)

B. *Kebutuhan Waktu*

Kebutuhan waktu merupakan hitungan per mili-detik untuk melakukan satu kali hasing pada sebuah *string* dengan menggunakan algoritma fungsi *hash*. Kebutuhan waktu diperlukan untuk memperkiraan lamanya waktu pemecahan *password* pengguna dengan menggunakan *brute force attack* pada algoritma fungsi *hash* tertentu. Tabel 5. dibawah menunjukkan waktu rata-rata *hashing string* dengan panjang 6 sampai 10 karakter pada beberapa algoritma fungsi *hash*.

Tabel 5. Waktu Rata-Rata *Hashing* pada Beberapa Algoritma Fungsi *Hash* (Sumber : Percobaan Pribadi).

No.	Algoritma Fungsi *Hash*	Ukuran Block Output	Waktu Rata-Rata *Hashing* String 6 – 10 Karakter
1.	GOST3411	256 Bit	2.4
2.	MD2	128 Bit	0.2
3.	MD4	128 Bit	< 0.1
4.	MD5	128 Bit	< 0.1
5.	RIPEMD 128	128 Bit	< 0.1
6.	RIPEMD 160	160 Bit	< 0.1
7.	RIPEMD 256	256 Bit	< 0.1
8.	RIPEMD 320	320 Bit	< 0.1
9.	SHA-1	160 Bit	0.5
10.	SHA 224	224 Bit	0.6
11.	SHA 256	256 Bit	0.6
12.	SHA 384	384 Bit	0.6
13.	SHA 512	512 Bit	0.6
14.	SHA-3	288 Bit	0.8
15.	SHA-3 224	224 Bit	0.8
16.	SHA-3 256	256 Bit	0.8
17.	SHA-3 384	384 Bit	0.8
18.	SHA-3 512	512 Bit	0.8
19.	Skein 256	256 Bit	< 0.1
20.	Skein 512	512 Bit	< 0.1
21.	SM3	256 Bit	0.5
22.	Tiger	192 Bit	< 0.1
23.	Whirlpool	512 Bit	1.5

C. *Banyaknya Percobaan*

Banyaknya percobaan merupakan hitungan matematis terhadap seluruh kemungkinan kombinasi *password* yang dapat digunakan oleh pengguna. Tabel 6. dibawah menunjukkan banyaknya kemungkinan percobaan yang harus dilakukan oleh pihak penyerang (dengan asumsi menggunakan kombinasi "abcde-fghij-kalmn-opqrs-tuvwx-yz" sebagai *password*nya).

Tabel 6. Banyaknya Seluruh Kemungkinan Percobaan.

abcdefghijklmnopqrstuvwxyz			
Karakter	**Susunan**	**Percobaan**	**Bilangan**
26	1	26	26
26	2	676	676
26	3	17,576	17 Ribu
26	4	456,976	456 Ribu
26	5	11,881,376	11 Juta
26	6	308,915,776	300 Juta
26	7	8,031,810,176	8 Milyar
26	8	208,827,064,576	208 Milyar
26	9	5,429,503,678,976	5 Triliun
26	10	141,167,095,653,376	141 Triliun

D. *Kebutuhan Memory*

Kebutuhan *memory* merupakan banyaknya *memory* (free space pada *hardisk*) untuk menyimpan hasil

serangan *dictionary attack*, umumnya hal ini dipertimbangkan untuk melakukan *log* atau pencatatan dan/atau persiapan melakukan serangan lainnya seperti *dictionary attack*. Tabel 7. dibawah menunjukkan kebutuhan *memory* seluruh kombinasi *password* (dengan asumsi menggunakan kombinasi "abcde-fghij-kalmn-opqrs-tuvwx-yz" sebagai *password*nya).

Tabel 7. Kebutuhan *Memory*.

Untuk 5 Susunan Kata pada "a - z"			
No.	**Algoritma**	**Panjang MD**	**Memory (Dalam GB)**
1	elf64	64	1
2	MD2	128	2
3	MD4	128	2
4	MD5	128	2
5	SHA-1	160	2
6	SHA-2 (224 bit)	224	3
7	SHA-2 (256 bit)	256	3
8	SHA-2 (384 bit)	384	5
9	SHA-2 (512 bit)	512	6
10	Tiger	192	2

Pada tulisan ini *dictionary attack* akan dilakukan pada *password* dengan panjang maksimal 5 karakter yang telah dihasing dengan menggunakan algoritma MD5. Pemilihan maksimal 5 karakter berkaitan dengan lamanya waktu yang diperlukan dalam menyusun hasil tulisan ini. Namun tidak menutup kemungkinan *dictionary attack* untuk dilakukan pada jumlah maksimal karakter yang lebih banyak atau menggunakan algoritma fungsi *hash* lainnya.

V. HASIL *DICTIONARY ATTACK*

Dictionary attack pada tulisan ini akan dilakukan dengan menggunakan *sampling password* yaitu huruf kecil 'q', kemudian dilakukan *hashing* terhadap *password* tersebut dengan menggunakan algoritma MD5, serta melakukan perbandingan waktu hasil serangan dengan serangan praktis lainnya yang telah penulis lakukan pada penelitian sebelumnya, yaitu *brute force attack*. Tabel 8. dibawah menunjukkan hasil *differential attack.*

Tabel 8. Hasil *Dictionary attack.*

Message Digest	**Hasil Brute Force**	**Waktu Rata-Rata (ms)**	
		BFA	**DA**
1 Karakter			
7694f4a66316e53c8cdd9d9954bd611d	q	0.50	0.625
2 Karakter			
099b3b060154898840f0ebdfb46ec78f	qq	8.50	0.625
3 Karakter			
b2ca678b4c936f905fb82f2733f5297f	qqq	312.20	0.625
4 Karakter			
3bad6af0fa4b8b330d162e19938ee981	qqqq	29,360.50	0.625
5 Karakter			
437599f1ea3514f8969f161a6606ce18	qqqqq	3,821,929	0.625

Berdasarkan pada tabel hasil *dictionary attack* diatas, dapat diketahui bahwa untuk dapat memecahkan *password* pengguna yang di*hash* dengan menggunakan algoritma MD5, *dictionary attack* dengan sistem pengkelasan memiliki waktu serangan yang jauh lebih cepat dibandingkan dengan *brute force attack*. Selain itu *dictionary attack* sistem pengkelasan juga memiliki waktu yang lebih terprediksi dalam menyelesaikan sebuah serangan, hal tersebut terjadi karena serangan pada *dictionary attack* sistem pengkelasan hanya melakukan perbandingan variabel *string* input *hashed password* pengguna dengan data *hashed password* yang tersimpan didalam *database* penyerang. Hasil serangan praktis ini sekaligus membuktikan, bahwa tidak terdapat keamanan *absolute* pada data/informasi yang disimpan maupun yang dikirim-terimakan, khususnya bagi negara atau organisasi yang memiliki perangkat, data, daya dan *source* komputasi serangan yang sangat besar.

KESIMPULAN

Teknik *dictionary attack* merupakan salah satu serangan praktis yang paling efektif untuk diterapkan dalam memecahkan *hashed password* pengguna, namun teknik tersebut membutuhkan biaya (perangkat, data, daya) dan khususnya *memory* yang sangat besar. Berdasarkan pada hasil percobaan dalam penelitian ini, dapat diketahui bahwa *dictionary attack* pada *hashed password* pengguna dapat dioptimalisasikan dengan menggunakan sistem pengkelasan, sehingga proses pencarian dan perbandingan dalam *dictionary attack* dapat lebih fokus ke hasil yang ingin dicapai, selain itu hasil serangan terhadap *hashed password* menjadi lebih cepat dan waktu serangan lebih terprediksi.

Masalah utama dalam melakukan serangan teknis, baik *brute force attack*, maupun *dictionary attack* adalah kebutuhan waktu dan *memory* yang sangat besar, namun seiiring dengan perkembangan perangkat komputasi dan media penyimpanan, maka serangan tersebut akan menjadi serangan yang lebih mudah untuk dilakukan, terlebih jika *super computer* dan *quantum computer* telah membumi dan memiliki harga yang lebih terjangkau layaknya komputer Turing saat ini.

REFERENCES

[1] Arief, M. Rudyanto. *Otentikasi Multi Faktor Untuk Meningkatkan Keamanan Komputer*. Teknik Informatika. STMIK Amikom. Yogyakarta.

[2] Indrajit, E. Richardus. *Manajemen Password*. pdf

[3] Maryanto, Budi. *Penggunaan Fungsi hash Satu-Arah Untuk Enkripsi Data*. Sekolah Tinggi Manajemen Informatika dan Komputer LIKMI. Bandung.

[4] Silalahi, Marzuki. Tumpal. Susanti, Hermina. *Perancangan Program Aplikasi Kriptosistem Menggunakan Algoritma Square yang Dimodifikasi dan Fungsi Hash SHA-1*. Universitas Tarumanegara. Jakarta.

[5] Suherman. *Bahan Ajar Pemrograman Database Delphi*. Cilegon.

[6] Prabowo, Imam. *Analisis Dictionary attack dan Modifikasinya Dalam Membobol Password Serta Solusi Melawannya*. Program Studi Teknik Informatika. Sekolah Teknik Elektronika dan Informatika. Bandung.

[7] *Terms of Condition. Tata Cara Pelaksanaan Pengadaan Barang/Jasa PT PLN (Persero) Secara Elektronik (e-Procurement)*.pdf

[8] *The Goverment of the Hong Kong Special Admnistrative Region*. 2008. *Password Management*.pdf.