Analisis Penggunaan Algoritma Bcrypt dengan Garam (*Salt*) untuk Pengamanan Password dari *Peretasan*

Muhammad Risqi Firdaus- 13520043[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*13520043@std.stei.itb.ac.id*

***Abstrak*—Seiring meningkatnya transaksi dalam jaring, meningkat pula jumlah kriminalitas di internet. Salah satunya peretasan. Salah satu langkah preventif menghindari peretasan ialah, menerapkan *hasing* pada penyimpanan password. Namun, tidak semua algoritma benar-benar aman dari berbagai metode peretasan. Dari algoritma *hasing* yang telah ditemukan, ada satu algoritma yang cukup efektif dan mudah diimplementasikan, yakni *salted bcrypt*. Algoritma ini dinyatakan cukup efektif untuk mengamankan password dari berbagai metode peretasan yang ada.**

Kata Kunci—Password, Bcrypt, Hash.

I. Introduction

Seiring bergesernya transaksi ke media online. Terjadi pula peningkatan kasus kriminalitasi di internet. Tahun 2021 saja, sudah berkali-kali terjadi kebocoran data. Ssalah satu data penting yang bocor ialah kata sandi.

Kemanan program menjadi hal yang sangat penting. Salah satu aspek pentiong dalam keamanan sebuah program ialah kemanan penyimpanan password. Menurut Bonsjak, dkk, 2019, autentikasi dengan pasword menjadi cara paling umum yang digunakan untuk mendapatkan akses kepada sebuah dokumen. Rahsia data pribadi perusahaan hingga pribadi sangat ditentukan sebarapa aman sebuah password disimpan[3].

Penyimpanan password pada *plain text* atau ala kadarnyasangat rawan dengan serangan peretas. Password pada *plain text* akan sangat mudah untuk didekrip. Akibatnya, data yang tersimpan menjadi lebih mudah untuk dicuri, bahkan bisa disalahgunakan untuk sebagai transaksi palsu.

Untuk menghindari hal tersebut maka diimplementasikanlah kriptografi dalam penyimpanan sandi. Sampai saat ini, sudah sangat banyak teknologi yang dapat digunakan untuk melakukan enkripsi penyimpanan sandi.

Seiring berkembangnya teknologi, muncul masalah pada kemanan sebuah produk hash atau hasil dari enkripsi, seperti *rainbow table, brute force,* serta *dictionary attack*. Selain itu, algoritma standar hashing dapat menghasilkan *cipher text* yang sama pada *plain text* yang sama. Hal ini menyebabkan algioritma hash, serta password lebih mudah ditemukan [2].

Pada sebuah algoritma hash standar, hasil hashnya dapat di-*brute force* dan dibandingkan dengan *rainbow table* serta *chiper text dummy* dari peretas sehingga jenis hash yang digunakan lebih mudah diterminasi. Akibatnya, hash yang digunakan tidaklah aman lagi [6].

Menurut Batubara, 2020, *brute force* menjadi algoritma paling naif yang sering digunakan untuk menjebol keamanan sebuah password. Namun, terdapat salah satu algoritma yang sangat sulit ditembus oleh *brute force*, yakni algoritma *bcrypt*. Algoritma ini disebut membutuhkan waktu lama atau bahkan tak hingga supaya bisa ditembus dengan *brute force*[2].

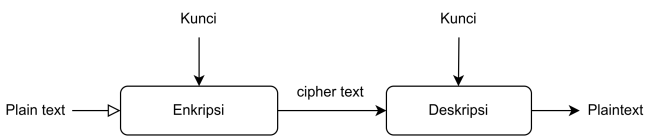
II. Teori Dasar

A. Kriptografi

Kriptografi secara istilah diartikansebagai ilmu yang mempelajari mengenai ilmu keamanan pesan[19]. Dalam buku *Applied Cryptography*, Bruce Scheiner menjabarkan bahwa kriptografi merupakan ilmu yang menjaga kemanan suatu pesan.

Menurut Suhardi, pada kriptografi digunakan beberapa istilah penting, sebagai berikut[17]:

1. Plaintext, berarti pesan asli.
2. Ciphertext, berarti pesan yang sudah dienkripsi.
3. Cipher, yang berarti algoritma yang dipakai untuk mengubah plaintext ke ciphertext.
4. Key, yang berarti kunci pada proses kriptografi yang hanya diketahui oleh pengirim dan penerima.
5. Encipher (Encrypt), yang berarti proses enkripsi plaintext ke ciphertext.
6. Decipher (Decrypt), yang berarti proses dekripsi ciphertext kembali ke plaintext.



Gambar 2.1 Proses enkripsi dan dekripsi

B. Hashing

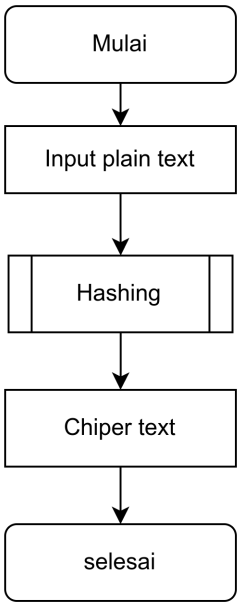
Hashing adalah proses fungsi searah yang mehasilkan sebuah kode dengan panjang sama, meskipun berasal dari nilai masukan yang panjangnya beda. Fungsi hash tidak pula dapat diinvers. Di dunia, Hash juga merupakan standar keamanan yang digunakan untuk mengamankan data [15].

Hashing dinilai lebih aman daripada enkripsi untuk pengamanan kata sandi karena fungsi hash akan menghasilkan string dengan panjang sama dan tak dapat diputarbalikkan ke

teks aslinya. Beberapa algoritma hashing di antaranya, SHA 1, SHA 2, SHA 256, dan SHA 512. Tabel 2.1, berikut, mengambarkan informasi perbandingan dari tiap algoritma hashing[18].

Fungsi hash harus memiliki tiga keamanan standar, yakni ketahanan akan benturan (hash yang berbeda untuk string yang berbeda), ketahanan pragambar kedua serta kethanan pragambar[13].

Proses pada hashing dapet dilihat sebagaimana tabel berikut[2],



Gambar 2.2 Alur proses hash secara umum.

C. Brute Force

Brute force merupakan teknik yang paling umum digunakan untuk meretas aplikasi website. *Brute-force* sangat efektif untuk mengambil akses server melalui peretasan password [9]. adalah teknik untuk menemukan kelemahan sebuah algoritma kriptografi yang dilakukan pada sistem keamanan komputer dengan mencoba semua kemuingkinan kunci satu persatu[18].

[I] Ada beberapa tool yang dapat digunakan untuk membantu peretasan dengan brute-force pada server seperti Hydra, Medusa dan Ncrack. Dengan menerapkan kakas di atas, peretas dapat dengan mudah mendapatkan akses, terlebih pada penyimpanan password tanpa hash atau dengan hash yang mudah ditebak. Beberapa cara yang dapat diaplikasikan untuk menghindari brute-force diantaranya, meningkatkan kompleksifitas password serta membatasi banyak percobaan permintaan akses.

D. Dictionary Attack

Prosedur pada *dictionary attack* cukup mirip dengan *brute-force*. Perbedaan kedua metode ini terdapat pada sumber yang digunakan. Jika brute-force menggunakan segala kemungkianan yang ada, *dictionary attack* menggunakan password berdasarkan kata-kata yang ada pada kamus. Pada sebagian sistem, *dictionary attack* dinilai kurang efektif sebab format password yang ada kini sudah sangat bervariasi[7].

Menurut Achmady, 2017, password dengan pola baru atau kombinasi huruf angka yang unik akan sulit untuk ditebak oleh kamus atau bahkan tak tertebak oleh sistem. Sistem hanya akan mengolah data berdasrkan data yang sudah dimiliki[16].

E. Rainbow Table Attack

Rainbow table merupakan kompilasi prekomputasi dari plainteks dan ketercocokan cipherteks. Penggunaan *rainbow tables* dapat mempercepat proses peretasan password dengan mencocokkan jenis hash yang digunakan. Sebagian besar *rainbow tables* bisa meretas hampir seluruh hash password yang ada[5].

F. Algoritma Bcrypt

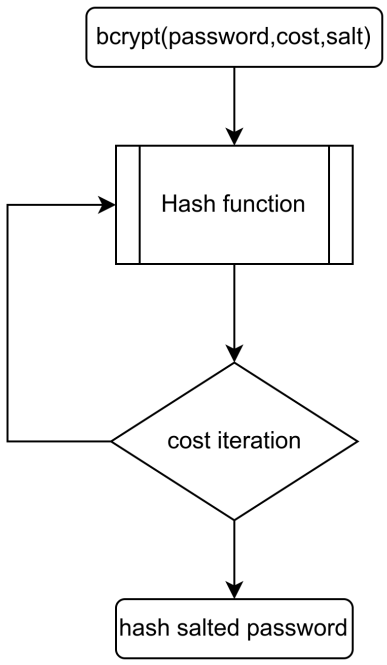
Bcrypt adalah fungsi hashing password dengan jumlah iterasi yang dapat ditingkatkan untuk menciptakan hash yang lebih aman dari serangan brute-force, tetapi akan memakan komputasi yang lebih besar[3].Ciri dari string hasil fungsi brcypt ialah memiliki awalan “$2a$” atau “$2b$” atau ”$2y$”.

Menurut Levent Ertaul, bcrypt adalah fungsi penurunan kata sandi yang dirancantg untuk sistem. Bcript merupakan algoritma yang paling populer dan tahan dari pertasan[11].

Bcrypt memliki algoritma yang mirip dengan fungsi blowfish block cipher. Bcrypt menggunakan prosedur EKSBlowfish untuk memperkuat enkripsinya, terutama untuk menghindari serangan brute force. EKSBlowfish juga di rancang dan digunakan untuk sebagai kata sandi kunci [12].

Pada alogritma Bcrypt, jumlah proses hash acak yang dijalankan disebut dengan cost. Cost memiliki jumlah minimal sebanyak 10 kali. Perulangan yang dilakukan pun dapat bervariasi hingga lebih dari 31 kali [6].

Menurut Jha dan Popli, dalam penelitiannya, membandingkan algoritma hash yang aman, dihasilkan bahwa algoritma bcrypt merupakan algoritma terbaik untuk solusi pengamanan sandi[8]. Berikut skema umum dari algoritma bcrypt,



Gambar

H. Salt (Garam)

Dalam kriptografi salt, atau garam, merupakan input random tambahan yang disimpan dan digunakan untuk memperkuat enkripsi atau hash dari sebuah password atau teks[84]. Pada metode ini, digunakan garam yang berbeda untuk tiap password, sehingga dihasilkan hash yang berbeda dari sebuah string yang sama.

Metode ini telah banyak digunakan untuk mengamankan password oleh banyak organisasi dan website. Pada aplikasinya, penempatan salt dapat ditaruh di awal maupunn diakhir dari hash. Masih diperlukan metode penempatan salt yang efektif sehingga terbentuk hash yang paling aman[8]. Skema salt sendiri dapat dilihat melalui fungsi berikut,

$$Hash(password, salt) \quad (2.1)$$

atau

$$Hash(password, salt) \quad (2.2)$$

III. PEMBAHASAN

A. Hasil Uji Peretasan Algoritma Hasing MD5 dan SHA-256

MD5 dan SHA merupakan algoritma paling umum yang digunakan untuk mengamankan data. Pada pengujian kali ini, akan dilakukan pengujian untuk algoritma di atas dengan metode *brute-force* serta *dictionary attack* Pengujian kali ini menggunakan dataset password dari Universitas Maribor, Slovenia, dengan besar dataset 185.643 password dalam MD5, SHA-1 dan SHA-256 oleh Bonsjak pada 2018[10].

Pada penelitian kali ini, Bonsjak, dkk, menggunakan dua komputer dengan spesifikasi berbeda. Pertama, *grapic card* AMD Radeon R9 280X, RAM 8 GB dan prosessor i5-4760k dengan kecepatan hash MD5 rerata 8.900MH/s. Kedua, 3 nVidia GeForce, RAM 64 GB dan prosessor i&-6700k dengan kecepatan hash kombinasi rerata 88.500 MH/s. Sedangkan, untuk kakas yang digunakan pada penelitian ini adalah hashcat.

Pada penelitian kali ini data dikumpulkan berdasarkan pola string yang membentuk password. Berdasarkan data data yang dihimpun, algoritma *brute-force* menghasilkan waktu retas rerata 2 menit 12 detik per password. Lebih lanjut untuk data peretasan tiap kategori dan waktu terdapat pada tabel berikut,

[2] no	[3] jenis serangan	[4] Jumlah	[5] %	[6] Waktu
[7]	[8] Brute-force	[9]	[10]	[11]
[12] 1	[13] Pola Lama(2 huruf kecil diikuti oleh 4 digit)	[14] 115.498	[15] 76,89	[16] 20 detik
[17] 2	[18] Kenaikan sampai panjang 6 (mixed alpha special num)	[19] 12.052.06	[20] 8	[21] 2 menit 12 detik
[22] 3	[23] Hanya digit dari panjang 7 sampai 12	[24] 656	[25] 0,44	[26] 3 menit 17 detik
[27] 4	[28] Kenaikan sampai panjang 8 (charset khusus huruf kecil)	[29] 7094	[30] 4,72	[31] 22 menit 6 detik
[32] 5	[33] Panjang 9 dan 10 (charset khusus campuran, pola khusus)	[34] 2071	[35] 1,38	[36] dibatalkan setelah lebih dari 30 menit
[37] 6	[38] Kenaikan sampai panjang 9 (charset bahasa asing)	[39] 36	[40] 0,03	[41] dibatalkan setelah lebih dari 15 menit
[42] 7	[43] Perbaikan Pola (mixed alpha special num panjang 8)	[44] 9701	[45] 6,46	[46] 10 jam 20 menit
[47] 8	[48] Panjang 10 huruf kecil alfabet bahasa inggris	[49] 67	[50] 0,05	[51] 49 menit 33 detik
[52] 9	[53] Panjang 10 huruf kecil alfabet bahasa inggris dan digit	[54] 222	[55] 0,15	[56] 20 jam
[57]	[58] Dictionary	[59]	[60]	[61]
[62] 10	[63] 674.096 kata kamus	[64] 248	[65] 0.17	[66] 1 detik
[67] 11	[68] 14.457.264 kata kamus	[69] 36	[70] 0.03	[71] 1 detik

Tabel 3.1 Tabel pengujian untuk password dengan komputer peneliti.

Berdasarkan data pengujian plain teks, tingkat kesuksesan peretasan mencapai 99.49 persem. Dengan menyisakan 953 dari 185.643 password yang ada.

Berdasarkan komputasi yang dilakukan, ditemukan pula perkiraan waktu komputasi dengan desktop pc biasa. Ditemukan bahwa waktu yang diperlukan untuk meretas dengan brute-force sistem enkripsi MD 5 ialah 10 jam 37 menit dan SHA-1 selama 21 jam 43 Menit. Sedangkan, dengan metode *Dictionary* waktu yang diperlukan utnuk meretas dengan metode brute-force ialah untuk algoritma MD5 ialah 6 detik, dan untuk SHA-1 selama 7 detik.

Fungsi Hash	Estimasi dengan PC biasa	
	Brute-force	*Dictionary*
MD5	10 jam 37 menit	6 sekon
SHA-1	21 jam 43 menit	7 sekon

Tabel 3.2 Perkiraan waktu komputasi dengan PC biasa.

Dari penelitian yang dilakukan pada bagian I, ditemukan bahwa algoritma hash biasa, seperti MD5 dan SHA-1 cukup mudah diretas dengan brute-force dalam waktu singkat. Selain itu, plainteks yang sama dapat menghasilkan hasil hash yang sama karena tidak memiliki nilai peubah unik sebagaimana keberadaaan salt dalam hash. Hal ini menyebabkan hash akan

sangat mudah untuk diretas menggunakan metode seperti dictionary dan rainbow tabel. Keberadaan hash yang tak unik dan rentan tabrakan akan menyebabkan proses peretasan menjadi lebih cepat karena terjadi timpang tindih hasil hash.

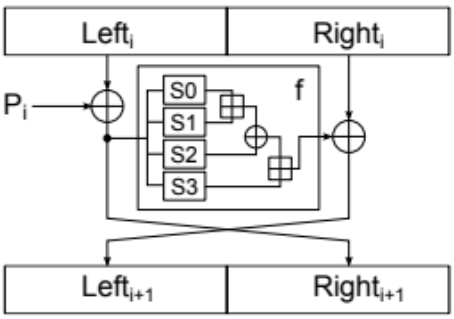
B. Pengamanan Password dengan Metode Salted Bcrypt

Telah diketahui bahwa bcrypt merupakan algoritma standar dalam pengamanan atau enkripsi password. Maka untuk meningkatkan keamanannya ditambahkan lah sebuah string random yang disebut garam. String garam nantinya akan disimpan sebagai pembanding dalam database. Keberadaan garam ini mengabitkan hasil hash yang berbeda pada password yang sama.

Untuk memahami keamanan dari algoritma salted bcrypt, perlu diketahui terlebih dahulu alur dari algortima tersebut. Untuk memahami programnya secara utuh, perlu dipugar program bcrypt terlebih dahulu. Sebagai dasar, fungsi EksBlowfishSetup merupakan fungsi dasar pada bcrypt,. Eksblowfish didesain untuk mengambil sandi masukan dari pengguna sebagai kunci dan menjaganya dari serangan. Sebagai dasar, digunakan algoritma blowfish [13] block chiper dari Scneider.

EksBlowFishSetup dimulai dengan memanggil initState, fungsi yang menyalin digit pertama dan dijadikan subkey. Selanjutnya EksBlowfishSetup akan memanggil Expandkey dengan masukan garam yang , kemudian dilakukan perulangan sebanyak 2 pangkat cost untuk melakukan ExpandKey terhadap garam dan key. Berikut algoritma dari EksBlowfish,

```
input: cost, salt, key
output: state
state ← initState()
state ← ExpandKey(state,salt,key)
repeat(2**cost)
        state ← ExpandKey(state, 0, salt)
        state ← ExpandKey(state,0,key)
return state
```

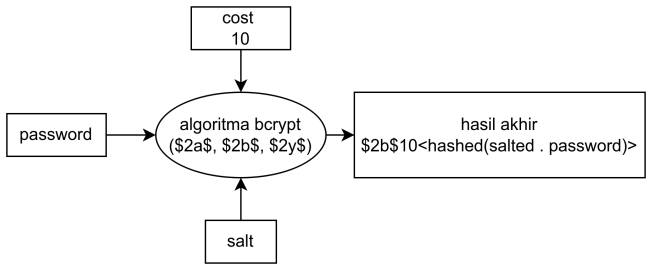


Gambar 3.2 Skema bitwise standar pada algoritma blowfish.
Sumber : IEEE [13]

Algoritma EksBlowfish tadi yang akan digunakan untuk memproses masukan pada bcrypt. Gambaran umum algoritma bcrypt dengan masukan nilai cost, garam serta password. Fungsi bcrypt nantinya akan mengembalikan konkatenasi dari hasil hash ketiganya. Berikut psedocode dari fungsi bcrypt, diambil dari [12].

```
input: cost, slat, key
output: hash
state ← EksBlowFishSetup(cost,salt,password);
ctext ← "OrpheanBeholderScryDoubt";
Repeat(64) begin
        ctext← EncryptECB(state,ctext);
        end
return Concantenate(cost,salt,password)
```

Agar lebih jelas berikut skeama bentukan (konkatenasi) hasil hash dari bcrypt dengan rumus (2.1)



Gambar 3.2 Skema format implementasi bcrypt.

Program di atas menerima masukan berupa password serta mendpatkan garam dari waktu instan. Hasil hased (*salted bcryp*) dikirima bersama salt ke dalam database. Berikut gambaran database yang digunakan.

Pw	Salt
$2y$10$T0cCtKDi51kw15bmQOzWNOc.sVQAfKZ2XFZMgmXQGn7nQxcblAFiO	1638776990
$2y$10$LDKFvufk.n7kUyrXKtQy.ea8oUYk3l.tekTrr1z33lZsj35YVUdmC	1638777306

Gambar 3.3 Skema penyimpanan password dan garam

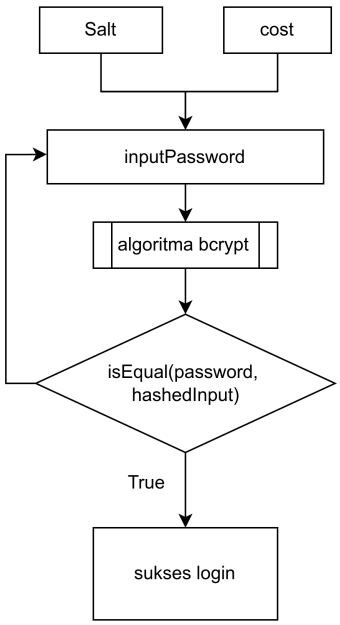
Salted bcrypt terdisiri atas komponen bcrypt, cost, salt serta key. Berikut contoh salted bcrypt serta, pembagiannya

$2<a/b/x/y>$[cost]$[22 character salt][31 character hash]
$2a$10$bOrOB/8n0Lf9M8DlYgF3teYWE
Lc/4xhv02m88G/hMTObwHYHDWT3i

1. Komponen kuning merupakan komponen jenis kode bcrypt.
2. Komponen merah menuinjukkan nilai cost dari bcrypt.
3. Komponen biru menunjukkan nilai dari salt yang digunakan.
4. Komponen hijau merupakan hasil hash dari key.

Setelah diketahui alur pembuatan serta penyimpanan data pada algoritma bcrypt, selanjutnya perlu diketahui skema verifikasi password dengan bcrypt.

Hasil hash antara konkatenasi password dari user serta salt dari database akan dibandingkan dengan nilai hash yang disimpan dalam database. Ketika hasilnya dinyatakan identik atau sama maka password ainggap benar. Metode ini ditengarai dapat mengamankan hasil hash dari perbandingan dengan metode *rainbow table*.



1. *Gambar 3.3*

1.

2. Berikut contoh pseudocode implementasi pendaftaran dan verifikasi password pada program nyata.

3.

```
Function Regis
Input: password,
output: hash,salt

input(password);
salt← time();
cost ← 10;
result ← hash(salt,cost,password);
return result,salt;

function loginVverifvication
Input: database:[username,pass,salt] , inputPassword,
username
output: boolean

if(isUsernameInDb)then
        pwsalt← inputPass+db.salt;
        if(verrify(pwsalt,db.pass)then
                return 1;
        else
                return 0;
return 0;
```

Dari penelitian yang dilakukan, ditemukan bahwa metode untuk meningkatkan ekamanan website dapat dilakukan dengan memberikan salt serta menggunakan hash berjenis bcrypt. Penggunaan random string salt dapat berupa nilai sembarang, pada contoh, nilai salt diambil dari nilai pendaftaran pengguna.

1. *C. Hasil PengujianPerforma*

Pengujian dilakukan untuk mengetahui pengaruh karakter uji dengan waktu proeses algoritm bcrypt serta ketahannya terhadap serangan *brute-force*. Adapun data didapatkan dari sumber literatur [2].

1. Pengujian Algoritma Bcrypt terhadap Waktu Uji

Pertama, waktu proses algoritma bcrypt. Berdasrkan uji coba pada komputer intel i7 gen 10th, NVDIA 230MX, RAM 12GB, proses melakukan algoritma bcrypt dengan bahasa php pada 8 karakter uji menunjukkan waktu proses di bawah 1000 ms. Sedangkan, pada penelitian yang dilakukan Toras, library bcrypt pada node js, memelurkan waktu proses 287 ms untuk melakukan bcrypt pada 5 karakter uji.

2. Pengujian *Brute-Force* pada Karakter Alphabet

Pengujian BruteForce terhadap karakter alphabet adalah proses uji peretasan dengan semua kemungkinan kata sandi dengan plain teks karakter alphabet. Berikut hasil pengujian terhadap karakter alphabet,

No.	Jenis	Jumlah karakter	hasil	waktu
1.	Alphabet	4	ditemukan	4 hari
2.	Alphabet	5	ditemukan	5 hari
3.	Angka	7	Ditemukan	8 jam
4.	Alpha numerik	7	Tidak ditemukan	5 hari

Tabel 3.3 Hasil pengujian salted bcrypt terhadap *brute-force*

Berdasarkan estimasi yang dilakukan bonsjak, dkk, 2019, dalam penelitannya, komputer biasa baru bisa memproses bcrypt lima karakter dengan *brute-force* setelah 10 tahun, dan baru bisa menyelesaikan dictionary attack password setelah 308 hari.

	Estimasi waktu pada PC biasa	
	Brute-force	*Dictionary*
Bcrypt(5 karakter)	Lebih dari 10 tahun	lebih dari 308 hari

[72] Tabel 3.4

[73]

Berdasarkan pengujian yang dilakukan pemberian salt dapat menyelamatkan hasil hash dari collison atau benturan hasil hash yang sama. Selain itu, konkatenasi hasil hash menyebabkan password hampir tidak mungkin diretas, karena peretas harus menebak salt yang random. Hal ini menyebabkan metode salted bcrypt menjadi metode standar pada pengamanan password. Password yang diamankan dengan salted bcrypt relatif hampir tidak mungkin dipecahkan dengan komputer biasa, pun dengan komputer *High-end*, salted bcrypt masih relatif sulit dipecahkan.

Pada kombinasi tertentu, salted bcrypt masih belum bisa dipecahkan dengan brute-force. Begitu juga dengan rainbopws table, hasil enkripsi yang random, serta penambahan garam menyababkan salt hampir tidak mungkin dicocokkan dengan elemen dari rainbow table, sehingga keamanannya dapat terjaga. Selain itu, salted bcrypt tidak mungkin diretas juga dengan dictionary, karena kombinasi dari enkripsinya bisa

sangat unik dari nilai kamus.

Perlu digarisbawahi, bahwa pola atau kombinasi dari password secara signifikan memengaruhi keamanan dari password tersebut. Pada uji brute-force yang dilakukan, terlihat bahwa password numerik sangat mudah untuk diretas. Sehingga perlu adanya perstandaran pola kata sandi.

IV. SIMPULAN

Dari analisis, serta riset literatur yang dilakukan, diperoleh hasil bahwa, salted bcrypt merupakan metode yang paling efektif dalam upaya preventif menghindari terjadinya peretasan. Salted bcrypt dinilai memenuhi tiga syarat standar sebuah fungsi hash. Keberadaan salted semakin memperkuat keamanan dari salted bcrypt. Salt yang berbeda akan menyebabkan perbedaan hasil hash pada key atau password yang sama.

Salted bcrypt sangat efektif untuk dijadikan metode pengamanan password. Metode ini dinilai hampir tidak mungkin untuk diretas dengan metode apapun. Meski begitu, pada pola tertentu, seperti numerik saja, alted bcrypt akan mudah untuk diretas. Sehingga perlu adanya perstandaran pola agar kemungkinan peretasan dapat diminimalisasi.

REFERENCES

[1] B. Shavers and J. Bair, "Cryptography and encryption," Hiding Behind the Keyboard, pp. 133–151, 2016.

[2] Batubara, Toras & Efendi, Syahril & Nababan, Erna. (2021). Analysis Performance BCRYPT Algorithm to Improve Password Security from Brute Force. Journal of Physics: Conference Series. 1811. 012129. 10.1088/1742-6596/1811/1/012129.

[3] Boonkrong, Sirapat & Somboonpattanakit, Chaovalit. (2016). Dynamic Salt Generation and Placement for Secure Password Storing. IAENG International Journal of Computer Science. 43. 27-36.

[4] Chester, John A., "Analysis of Password Cracking Methods & Applications" (2015). Williams Honors College, Honors Research Projects. 7. https://ideaexchange.uakron.edu/honors_research_projects/7 [diakses tanggal 5 Desember 2021.]

[5] D. Arias, "Hashing in action: Understanding bcrypt," Auth0, 25-Feb-2021. [Online]. Available: https://auth0.com/blog/hashing-in-action-understanding-bcrypt/. [Diakses pada: 07-Dec-2021].

[6] E. Conrad, S. Misenar, and J. Feldman, "Domain 1: Access control," Eleventh Hour CISSP, pp. 1–21, 2014.

[7] G. Jha and N. Popli, "Method for Storing User Password Securely," IITM Journal of Management and IT, vol. 6, no. 1, pp. 95–99, 2015.

[8] Grover, Varsha and , Gagandeep, An Efficient Brute Force Attack Handling Techniques for Server Virtualization (March 30, 2020). Proceedings of the International Conference on Innovative Computing & Communications (ICICC) 2020, Available at SSRN: https://ssrn.com/abstract=3564447 or http://dx.doi.org/10.2139/ssrn.3564447. [Diakses pada tanggal: 14 Desember 2021]

[9] L. Bošnjak, J. Sreš and B. Brumen, "Brute-force and dictionary attack on hashed real-world passwords," 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2018, pp. 1161-1166, doi: 10.23919/MIPRO.2018.8400211.

[10] L. Ertaul, V. Gudise, and M. Kaur, "The 2016 International Conference of Wireless Networks," in World Congress on Engineering, 2016.

[11] N. Provos and D. Mazieres, "1999 USENIX Annual Technical Conference," in THE ADVANCED COMPUTING SYSTEMS ASSOCIATION, 2016.

[12] P. Gauravaram, "Security Analysis of salt||password Hashes," 2012 International Conference on Advanced Computer Science Applications and Technologies (ACSAT), 2012, pp. 25-30, doi: 10.1109/ACSAT.2012.49.

[13] PROVIDING PASSWORD SECURITY BY SALTED PASSWORD HASHING USING BCRYPT ALGORITHM VOL. 10, NO. 13, JULY 2015 ISSN 1819-6608 ARPN Journal of Engineering and Applied SciencesP. Sriramya and R. A. Karthika

[14] Quyuh H Dang. Secure Hash Standard. Technical Report, 2015.

[15] S. Achmady, "Analysis dictionary attack Dan Modifikasi Password cracking serta Strategi Antisipasi," Jurnal Sains Riset, vol. 7, no. 1, 2017.

[16] Suhardi, "APLIKASI KRIPTOGRAFI DATA SEDERHANA DENGAN METODE EXLUSIVE-OR (XOR)," Jurnal Teknovasi, vol. 3, no. 2, pp. 23–31, 2016.

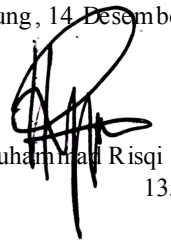
[17] W. Stallings and L. Brown, Computer security: Principles and practice. New York, New York: Pearson, 2018.

[18] W. Suadi, "MPLEMENTASI KRIPTOGRAFI DAN STEGANOGRAFI PADA MEDIA GAMBAR DENGAN MENGGUNAKAN METODE DES DAN REGION-EMBED DATA DENSITY," ITS-Undergraduate-Repository, Jul-2011. [Online]. Available: http://digilib.its.ac.id/public/ITS-Undergraduate-16398-5107100055-Paper.pdf. [Diakses pada tanggal: 10-Dec-2021].

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, 14 Desember 2021



Muhammad Risqi Firdaus
13520043