International Journal of scientific research and management (IJSRM)
||Volume||4||Issue||06||Pages||4347-4350||2016||
Website: www.ijsrm.in ISSN (e): 2321-3418

Enhancing Security in user authentication through honeyword

Ms. Arati A. Gadgil, Prof. S. D. Khatawkar
ADECT, Ashta ME CSE (Appearing)
arati.gadgil1@gmail.com
ME CSE
ADECT, Ashta
shriharikhatawkar@gmail.com

Abstract: Honeyword system used to detect password file disclosure. For each user set of honeyword is generated. When adversary have a password file, then it get confused which one is real password in honeyword set. Adversary enters all honeywords in the set. When honeywords are entered notification will be send to the admin. Author gives hybrid method for generation of honeyword. Hybrid method provides strong DoS resistance and flatness.

Keywords: authentication, login, password cracking, honeyword

I Introduction

System is said to be secure if two issues are concerned. First one is use of strong password policies and second is system must detect attacks like password file disclosure before getting any harm to the system. [1] Generally users chose their password which is easy to remember and uses single password for multiple system.

Disclosure of password file affected many companies like Yahoo, RockYou, Linkedln, eHarmony and Adobe [2] [3]. Website uses weak storage method to store password. Such as use of SHA-1 algorithm without salt and MD5 hashes. When adversary have password file then, by using some password cracking algorithm it is easy to invert hashed password in plaintext. [4] Implementing some policies like encrypt password using strong encryption algorithm, use of salt and use of key stretching or slow algorithms to increase password cracking time will minimize potential damage caused by password leak. Honeypot is one methods to identify occurrence of a password database breach. [5]

Proposed system uses SHA-1 whit salt for encryption of password. Honeywords are decoy passwords generated from original password. Proposed system uses hybrid method for generation of honeyword. Also system will check correlation between username and password. If username and password are co related then it becomes easy for adversary to identify correct password between set of honeyword.

II Honeywords

Honeywords are decoy passwords generated using two types of method. First one is legacy-UI method and second one is modified-UI method. [6] In legacy-UI method user is not aware about honeyword system. In legacy-UI user can get idea some security policies are implemented. Legacy-UI consists of chaffing with tweaking, chaffing with password model and chaffing with toughnut methods where modified-UI consists of tweak a tail method.

1 Chaffing by tweaking digit

In chaffing by tweaking method, honeywords are generated by tweaking the characters in passwords. System predetermines the value of t that is how many characters of passwords are replaced by randomly selected digit.

e.g. For password abc123$ honeywords are

abc1693 abc1736 abc1235

2 Chaffing with password model

In chaffing with password model honeywords are generated using same syntax of true password. [7] Password is "cat6light" then there are 3 letters, 1 digit and 5 letters, represented as L3+D1+L5. These characters are replaced by same syntax, here first 3 letters are replaced by random 3 letters then one digit replaced by random one digit and again 5 letters are replaced by 5 random letters. Therefore cat6light is replaced by fit1rings.

3 Chaffing with "toughnut"

Tough nuts are some special honeywords. Tough nuts are inserted by system at any random position in password. Inverting hash value of tough nuts is computationally infeasible. e.g. of tough nut is '9.50Pee[kv.0]!nwt'. Number and position of toughnut is selected randomly. User usually select passwords that are simple so tough nuts are complex. Therefore adversary can skip such honeyword for classic attacks.

4. Tweak a tail

In tweak a tail method when user choose their password then system will generate one random string and displayed to user. Generated string will be appended to the user password and this new string is a password of user.

III System implementation

1 User registration

User must be registered to the system. Generally user chooses password which are simple and easy to remember. [8] [9] Such type of password can easily identify using some basic attacks. [10] [11] [12] System allow user to select password which consist of minimum 8 characters, one digit and one special symbol is implemented. Also system will check correlation in percentage, between username and password. If correlation is greater than 50%, then system will force user to change the password.

Registration process is completed when user enters correct CAPTCHA. CAPTCHA is used to identify user is human or not. CAPTCHA protects system from spam and abuse. [13] After successful completion of registration, honeywords are generated using hybrid method.

2 Hybrid method

Proposed system uses hybrid method for generation of honeyword. Hybrid method is combination of other methods. In hybrid method first system will apply chaffing by tweaking digit method on password. In this step last t position will be tweaked. In second step output of first step is taken as input and system will apply chaffing by password model method.

Algorithm for hybrid method are given below,

Hybrid method Algorithm

```
1:  Procedure HybridMethod(P)
2:  a, b constants
3:  s ← lenghth(P)
4:  for I ←0 to a do
5:          Tweak(P)
6:          for j ←0 to b do
7:                  Split(P)
8:                  Honeyword ← P
9:          end for
10:   end for
11: end Procedure
```

Tweak method

```
1:  Procedure Tweak(P)
2:  R(D)← return digit
3:  d ← length(P)
4:   j←d-2
5:  for d to j do
6:          p[d]←R(D)
7:          j=j-1
8:  end for
9:  end Procedure
```

Split method

```
1:  Procedure Split(P)
2: R(D)← return digit
3: R(L) ← return letter
4: R(SS) ← return special symbol
5: d ← length(P)
6: for i=0 to d-1 do
7:          if P[i]==Letter then
8:                  P[i]←R(L)
9:          else if P[i]== Digit then
10:                 P[i]==R(D)
11:             else
12:                 P[i]←R(SS)
13:         end if
14: end for
15: end Procedure
```

3. Management of password

For each user account honeywords are generated. [14] Honeywords are encrypted using hash function with salt. Adding salt increases difficulty level for inversion process to the plain text. Salt is random string generated using Cryptographically Secure Pseudo-Random Number Generated (CSPRNG). Salt is unique for each user.

In proposed system original password is stored along with honeywords at any random index. System will stores user id and true password index in honeychecker. Password storage in honeyword system is shown in fig.1

User	Honeywords
u_1	$H(W_{1,1})$, $H(W_{1,2})$, **$H(W_{1,3})$**, $H(W_{1,4})$, $H(W_{1,5})$.....
u_2	$H(W_{2,1})$, $H(W_{2,2})$, $H(W_{2,3})$, $H(W_{2,4})$, **$H(W_{2,5})$**.....
u_3	$H(W_{3,1})$, **$H(W_{3,2})$**, $H(W_{3,3})$, $H(W_{3,4})$, $H(W_{3,5})$.....
u_4	$H(W_{4,1})$, $H(W_{4,2})$, $H(W_{4,3})$, **$H(W_{4,4})$**, $H(W_{4,5})$.....

Figure1 Password storage in honeyword system

4. Mechanism of Password Detection

System uses hybrid method for generation of honeywords. Therefore each user consists of set of honeywords. Honeywords are stored in hashed form along with original password. System implements mechanism for password detection. The purpose for implementing this mechanism is, as original password is stored along with honeyword therefore system can authenticate user. Honeychecker stores original password index.

When user login request arrives first system will check this user exists or not. If user exists it will check entered password is honeyword or not. If entered password is not honeyword then system will simply deny the access. If

the password is honeyword then system will check the index of that honeyword. As original password index is stored in Honeychecker, system will compare the index value. If the index value of entered honeyword and index of original password is same then user is registered user. So system will give access to that user. If index of entered honeyword and index of original password is not same then notification is send to the admin.

III Experiment and Result

1. Encryption of password.

Existing system uses hash function to protect user password. Hash algorithms are one way functions. They turn any amount of data into fixed length "finger" print that cannot be reversed.

They also have the property that if the input changes by even a tiny bit, the resulting hash is completely different. But there are many ways to recover passwords from hashes very quickly.

There are several easy-to-implement techniques that make these "attacks" much less effective. The two most common ways of guessing passwords are dictionary attacks and brute-force attacks. [14]

Dictionary Attacks

Trying apple: fail
Trying blueberry: fail
Trying 1234567: fail
.....
Trying letmein: fail
Trying qwerty: Success

A dictionary attack uses a file containing words, phrases, common passwords, and other strings that are likely to be used as a password. Each word in the file is hashed, and its hash is compared to the password hash. If they match, that word is password.

Brute Force Attacks

Trying aaaa: fail
Trying aaab: fail
Trying aaac: fail
Trying aaad: fail
…..
Trying acdb: fail
Trying acdc: Success!

A brute-force attack tries every possible combination of characters up to given length. These attacks are very computationally expensive, and are usually the least efficient in terms of hashes cracked per processor time, but they will always eventually find the password. System randomizes the hashes by appending random string, called a salt to the password before hashing. Salt is unique for each user. As shown in example below, this makes the same password hash into a completely different string every time.

In implemented system salt is generated by Cryptographically Secure Pseudo-Random Number Generator (CSPSNG). As the name suggests, CSPRNGs are designed to be cryptographically secure, meaning they provide a high level of randomness and are completely unpredictable. Therefore implemented system makes the total hash inversion process harder for an adversary in getting the passwords in plaintext from a leaked password hash file.

2. Dos resistance

Consider that adversary created few accounts intentionally. Therefore adversary knows true password of few accounts. Consider there is N no of user in the system. Adversary created m no of accounts. Now adversary has m no of true password. The probability that adversary can be entered honeyword and can be identified is calculated using following formula.

$$P = \frac{(N-M)(k-1)}{Nk}$$

Where k is no of honeywords.

Consider, N=1000, k=10 and m=10, then p=0.89

Probability that adversary can be caught is 89%, that is system providing strong DOS resistance.

3. Flatness

Implemented method satisfies perfect flatness as long as the correct password is not correlated with username. If username and password are correlated then original password is easy to guess. In system investigation of target user profile gives no advantage to an adversary in password guessing.

System also checks that entered username and password is correlated or not. If correlated then system force user to choose the password which is not correlated. Consider example user enters username as "Amruta" and password is "Amruta@1" then system calculates parentage correlation between username and password. If it is greater than 50 % then user must be change their password. In this example correlation is 75 %. Therefore user must change their password. Therefore system achieves strong flatness.

4. Storage cost

Compute the storage requirement of implemented system, a typical password file system requires hN plus storage for usernames, where N stands for the number of users in the system and h denotes length of password hash in bytes. On the other hand this is khN where k denotes the number of the sweetwords assigned to each account. For our approach we assume that each index requires 4 bytes and the storage cost becomes 4kN + hN + 4N. As no of honeywords increases storage cost for system increases. But goal is to make system secure which can be achieved by generating

honeyword. Table 5.1 shows comparison of honeyword generation method.

Sr. No	Honeyword method	DOS resistance	Flatness	Storage cost
1	Tweaking	Weak	weak	hN
2	Password model	Strong	strong	hkN
3	Hybrid method	Strong	strong	4kN +hN + 4N

Table 1 Comparison of honeyword generation method

IV Conclusion

Proposed system identifies password file disclosure before getting harm to the system. Use of ReCAPTCHA protects system from spam and abuse. Hybrid method which combines strength of others method is used for generation of honeyword, which provides strong DOS resistance and also flatness but increases storage cost. As no of honeywords increases storage cost of system will increase. Passwords are encrypted using hash function with salt which makes inversion process difficult for attacker.

In future, system can use honeyword mechanism to detect theft and gathering information about their source, attack patterns, final target and purpose of attacker by using fake interactive sessions.

V Reference

[1] Imran Erguler , Achieving Flatness: Selecting the Honeywords from Existing User Passwords, IEEE Transactions on Dependable and Secure Computing 2015.

[2] D.Mirante and C. Justin, Understanding Password Database Compromises, Dept. of Computer Science and Engineering Polytechnic Inst. of NYU, Tech. Rep. TR-CSE-2013-02, IEEE, 2013.

[3] I. Paul. Update: LinkedIn confirms account passwords hacked. PC World, 6 June 2012.

[4] K. Brown, The Dangers of Weak Hashes_SANS Institute InfoSec Reading Room, Tech. Rep., ,2013.

[5] M. H. Almeshekah, E. H. Spafford, and M. J. Atallah, Improving Security using Deception, Center for Education and Research Information Assurance and Security, Purdue University, Tech. Rep. CERIAS Tech Report 2013-13, 2013.

[6] A. Juels and R. L. Rivest, Honeywords: Making Password cracking Detectable, in Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, ser. CCS13. New York, NY, USA: ACM, 2013, pp. 145160. [Online].Available:http://doi.acm.org/10.1145/2508859.2516671

[7] H. Bojinov, E. Bursztein, X. Boyen, and D. Boneh, "Kamouflage: Loss-resistant Password Management," in Computer Security-ESORICS 2012. Springer, 2010, pp. 286-302

[8] P.G. Kelley, S. Komanduri, M.L. Mazurek, R. Shay, T. Vidas, L. Bauer, N. Christin, L.F. Cranor, and J. Lopez. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In IEEE Symposium on Security and Privacy (SP),2012.

[9] A. Vance, If Your Password is 123456, Just Make It Hackme, The New York Times, vol.20,2010.

[10] J. Bonneau and S. Preibusch, The Password Thicket: Technical and Market Failures in Human Authentication on the Web, in WEIS, 2010.

[11] M. Bakker and R. van der Jagt. GPU-based password cracking. Technical report, Univ. of Amsterdam, 2010.

[12] Y. M. Weir, S. Aggarwal, B. de Medeiros, and B. Glodek, Password Cracking Using Probabilistic Context-Free Grammars, in Security and Privacy, 30th IEEE Symposium on. IEEE,2009, pp.,391- 405.

[13] L. V. Ahn, M. Blum, N. J. Hopper, and J. Langford, CAPTCHA: Using Hard AI Problems for Security, in Proceedings of the 22nd International Conference on Theory and Applications of Cryptographic Techniques EUROCRYPT03, ser. Lecture Notes in Computer Science, vol. 2656. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 294311.

[14] F. Cohen, The Use of Deception Techniques: Honeypots and Decoys, Handbook of Information Security, vol. 3, pp. 646655, 2006.

[15] C. Herley and D. Florencio, Protecting financial institutions from brute-force attacks, in SEC08, 2008, pp. 681685.