Improved Hashing and Honey-Based Stronger Password Prevention against Brute Force Attack

Khin Su Myat Moe
Department of Computer Engineering and Information Technology
Yangon Technological University
Yangon, Myanmar
myatmoe.66@gmail.com

Thanda Win
Department of Computer Engineering and Information Technology
Yangon Technological University
Yangon, Myanmar
thanda80@gmail.com

Abstract – **Nowadays, security of password file is one of the most important problems for millions of users and companies in various fields. So, many systems store the password files in database using the various hashing and salting algorithm. However, password hashing is not secure by attackers because they try to get user's password in password file that are stored in the database using various attacks such as brute force attack, password guessing attack, etc. Therefore, some systems store the passwords in the database with honeywords or fake passwords using honeywords generation algorithm to prevent the attacks from hackers. But the current existing system using honeywords generation algorithm meets the two problems. The first problem is typo safety problem and the last problem is storage overhead problem. In this paper, we propose a novel honeyword generation approach which decreases the storage overhead, typo safety problem and also reduces the other drawbacks of existing honeywords generation techniques such as old password management problem, etc. Our proposed system stores the other users' passwords as honeywords in the database instead of creating the honeywords for reducing storage overhead problem and typo safety problem. Moreover, we store the password and honeywords into the database using a unique hashing algorithm with very low time complexity as most of the steps involved simple binary operations.**

Keywords – password file, hashing and salting algorithm, password hashing, storage overhead problem, honeywords generation algorithm

I. INTRODUCTION

The maintenance of password file in the database becomes the main challenge in various areas because many real world systems choose password based encryption algorithm. So, the password files play an important role in millions of users and companies such as Yahoo, RockYou, LinkedIn, eHarmony and Adobe [1],[2] since leaked password makes the user target of many possible cyber-attacks. Since, many companies try to protect the password files using hashing and sating algorithms. For example, LinkedIn passwords were using the SHA-1 algorithm without a salt and the eHarmonly passwords were also stored using unsalted MD5 hashes [3]. If the password file is attacked by attacker using password cracking techniques, the attackers can easily get the password files.

Honeyword generation method is one of the methods to defense against the stolen password file from attackers. In this approach, the system stores the list of password which contains the real user's password with honeywords from honey generation algorithm. Hackers who steal databases of user logins and passwords only have to guess a single correct password in order to get access to the data. When the database or password file becomes readable by using brute force attack, the attackers get the secrete passwords. To speed up the process, attackers have access to sophisticated software that can send thousands of passwords each minute to applications in an attempt to decrypt the data. Using higher speed, multi-core processors also shortens the time it can take to break encryption. With HE, decrypting with an incorrect password results attempt. For example, if a hacker made 100 password attempts, they would receive 100 plain text results. Even if one of the passwords were correct, the real data would be indistinguishable from the fake data [4].

The core innovation of the honeywords generation scheme is to store the user's real password with honeyword and the classification of honeywords and user's real password. The current existing honeywords generation method has weakness in password storage and old password management problem. So we propose the following facts to overcome vulnerability of the existing system.

- We propose honeywords generation methods to reduce storage overhead problem, typo safety problem and our proposed method is flatness.
- Moreover we propose hashing and salting algorithm with very low time complexity for securing stored passwords.

The rest of this paper describes as follows. In section II and III, we discuss related work and research methodology respectively. We present system flow of proposed method in section IV and section V presents comparative study on honeywords generation method. Section VI describes experimental result for the hashing algorithm and finally we conclude the paper.

II. RELATED WORK

Ziya Alper Genc, Suleyman Kardas, Mehmet Sabir Kiraz discussed that password is easy to crack with the improvement

978-1-5386-2778-5/17/$31.00 ©2017 IEEE

in the graphical processing unit (GPU) technology. An attacker can recover a user's password using brute-force attack on password hash. Once the password has been recovered no server can detect any invalid user authentication [6].

Ari Jules and Ronald L. Rivest described that how honeyword was produced and this honeyword was stored with real user password in password file. The password file is attacked by the hackers using brute force attack and this system can deceive to the hackers. This method can be caused storage overhead and typo safety problem. And then, they also discussed the storage of the password file and key expansion for covering brute force attack. Revealing the password file by the attacker is a serious security problem that has affected many users and company like LinkedIn, Yahoo, and eHarmony [7].

Many technicians considered how to reduce storage overhead problem and they created many methods. Among them, the new honeywords generation also called honey circular list or paired distance protocol (PDP) method was described by Nilesh Chakraborty and Samrat Mondal. That method can reduce storage overhead problem compared to the earlier existing methods. But the storage overhead problem has still remained in honey encryption process [8].

Many password systems, particularly for government and industry users, store hashes of users' old passwords usually the last 10 [9]. When a user changes her password, she is prohibited in such systems from reusing any stored ones. The related proposal by Schechter, Herley, and Mitzenmacher [10], which relies on a similar data structure called a "count-min sketch," allows one to reject new passwords that are already in common use within a user population. A better option is to not store old passwords on a per-user basis.

The example of existing system flow design of honeywords generation algorithm is shown in following fig. 1.

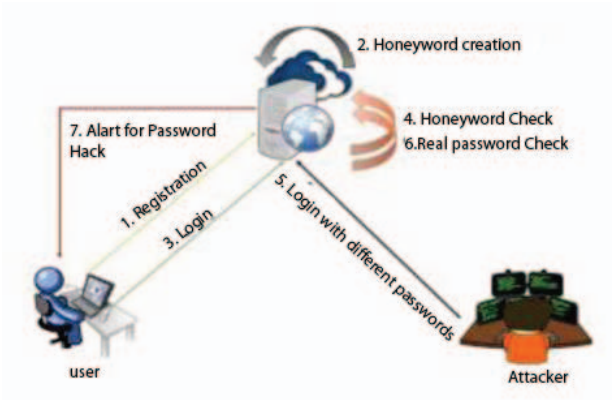


Fig. 1. Existing System [5]

The existing system flow includes seven steps in the above figure 1. Firstly, if the user is a new user, the user makes the registration process. After making the registration process, the system creates honeywords and honeyword indexes for password file protection. On the other hand, the user makes login process. When the user tries to login with his username and password, the honeychecker checks the entire password is the real password or honeyword. If the user has real password, the system allows this user to enter the system. Otherwise, the server sends an alarm message to the system administrator for entering honeywords. However, there are some limitations in honeywords generation process such as storage overhead problem, old password management problem and so on.

III. RESEARCH METHODOLOGY

The main idea of this section is to overcome storage overhead problem and old password management problem in honeywords generation method. Moreover, we describe case analysis of hashing and salting algorithm for securing password file.

A. Honeywords Generation Method

Passwords are notoriously weak authentication mechanism because users frequently choose poor and repeatedly passwords. The attacker can easily know this poor password. So, the system stores the correct password with several honeywords for each account in the database to deceive attackers. Honeywords also called decoyed passwords are used to detect attack against hashed password database. However, instead of generating honeywords and stored them in the password file, we use the existing user passwords as honeywords. In order to achieve this, the existing password indexes for each account which we called honeyindexes, are randomly assigned to a newly created account of user if the new account is created. And then, if a new user that creates new account, he gets randomly index number. The correct hashing password for this account is kept with this index number in the password list.

The main purpose of our method consists of two modules. The first module is to reduce storage space compared to the existing honeywords generation methods. The second module is that our method don't need to consider about typo safety problem. By using the passwords of other users as honeywords, the attackers become more complicate which password is fake password or which password is correct.

For each user account, we create two password tables in the database. The first table is stored into the main server and the second table is stored into the honey checker. Honey checker is an auxiliary secure server to assist with the use of honeywords. The honey checker can save store secret information and can raise an alarm to the administrator when an irregularity is detected. Table I has two elements: first element is the username of the account and the second is the honeyindex set for the representative account. The table is sorted randomly according to the user enter position. There are also two elements in the second table. The first one is the real password index of the account and the next is hashing of the corresponding real password. The table is sorted randomly according to the user registration position.

TABLE I
Example password files F1 in main server

Username	Honeyindex Set
John	(2,3,4,5)

Marry	(1,3,4,5)
Smith	(1,2,4,5)
Joy	(1,2,3,5)
Herry	(1,2,3,4)

TABLE II
Example password files F2 in Honey checker

Correct index	Hashing password
1	D7F6A6763403E357C
2	A946A6763403876
3	E7F6A4567403E9A1
4	F6A67633333357C
5	C1234T7F6A67634

Fig.2 shows the flowchart for the verification process of honeywords and real passwords. The main purpose of the honeywords generation algorithm is to hide the real password with many fake passwords.

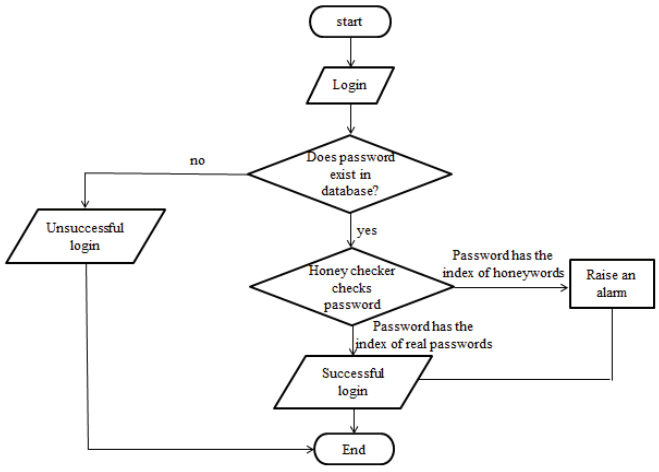


Fig. 2 Flowchart for Verification of Honeywords and Real Passwords

When a user sends a login request with password to the server, the server checks this user's password exists or not in the database. If password doesn't exist in password list, the server returns unsuccessful login to user. On the other hand, the server sends password to honey checker for classification of honeywords and real password if the password is in the password list. If the password is honeyword, the system raises an alarm to system administrator. When password is real password for this user account, the system allows this user to enter into the system.

B. Hashing and Salting Algorithm

In our proposed system, hashing and salting algorithm is considered to provide better security for key or password with faster time. This algorithm is as follows:

Step 1: Convert user's password into binary string.
Step 2: Adding padding bits to this binary string.
Step 3: Making flapping of the binary 1's and 0's in string.
Step 4: Perform XOR operation with salting binary string and string formed by combination of zero's and one's
Step 5: Rotate the string left or right by r character depending on the system.
Step 6: Convert the binary string into hexadecimal string.

C. Case Study I

Example- user selected password: hello123
Step1: Convert the input expression (password: hello123) into binary string.
01101000 01100101 01101100 01101100 01101111 00110001 00110010 00110011

Step 2: Padding bits are added a random string of character before making password hashing.
00000000 01101000 01100101 01101100 01101100 01101111 00110001 00110010 00110011

Step 3: Flapping of the binary 1's and 0's in the string.
11111111 10010111 10011010 10010011 10010011 10010000 11001110 11001101 11001100

Step 4: Performs XOR operation of binary strings with an equal size string formed by combinations of X zero's and one's. In this case, the value of X is 4. It means String2 is formed with 4 zero's and 4 one's.
String1:
11111111 10010111 10011010 10010011 10010011 10010000 11001110 11001101 11001100
String 2:
00001111 00001111 00001111 00001111 00001111 00001111 00001111 00001111 00001111
After XORing,
11110000 10011000 10010101 10011100 10011100 10011111 11000001 11000010 11000011

Step 5: Rotates the string right by r characters. Here, the value of r is 4. So, first 4 characters are rotated to the right.
00111111 00001001 10001001 01011001 11001001 11001001 11111100 00011100 00101100

Step 6: Convert the binary expression into hexadecimal string.
The resulting password of hexadecimal string (3f98959c9c9fc1c2c) is stored into the database.

IV. PROPOSED SYSTEM FLOW

In this system, there are two main parts: the first part is new user registration process and the last part is member login

process. The flowchart for the proposed system is shown in fig. 3. In this flowchart, the user makes the registration process if the user is new user. Otherwise, the user can make the login process. If the registration process successful, the system creates honeywords for this user's password. And then, this user's real password and honeywords are stored into the database using hashing and salting algorithm. If the user is a member of the system and he tries to enter into system, the system checks whether this user's password exists into the database or not. If password doesn't exist in the database, the server returns unsuccessful login message is released to this user. If this password exists in the database, the server sends this user's password to honeychecker for classification of real password and honeyword. If the password is real password, the honeychecker allows this user to enter into the system. Otherwise, the honeychecker sends an alarm message to administrator for entering the honeywords into the system.

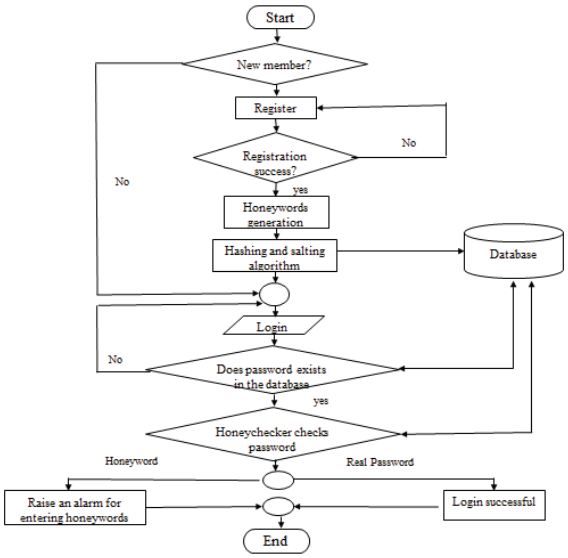


Fig .3: Flowchart for the proposed system

V. COMPARATIVE STUDY ON HONEYWORDS GENERATION METHODS

In this section, we describe the comparison of the honeywords generation methods including our proposed models in terms of flatness, typo safety and storage overhead.

A. Flatness

If the system maintains k sweetwords against a user u_i then attacker may get confused among k possible options once list of W_i is compromised. The list of Wi contains one sugarword (the real password) and (k-1) honeywords. Now it may happen that t adversary can easily identify t password chosen by the user from the list W_i (e.g if there exists a correlation between username and password). A honeyword generation algorithm is said to be perfecty-flat adversary has no advantage while identifying the user's original password from the list of W_i. If the honeyword generation algorithm is perfectly flat then the probability of selecting the original password of user from list W_i is 1/k. If the probability of selecting user password from the list W_i is slightly greater than 1/k, then the honeyword generation algorithm is approximately-flat. A good honeyword generation algorithm is required to be perfectly-flat [9].

The proposed honeywords generation method is a good honeywords generation method because the probability of selecting the original password of user from password file is 1/k. So, our proposed system is perfectly flat because we use the other user's passwords as honeywords.

B. Typo Safety

A honeyword generation technique is called typo safe if typing mistake of users during entering of the password does not get match with any of the honeywords [9].

Our method is better than the existing method because we choose the other user's password as honeywords. So, we can mostly reduce the typo safety problem because user cannot misunderstand with other user passwords.

C. Storage Overhead

Using the existing honeyword generation algorithms maintains k-1 extra passwords along with the original password of user, in the password file F. On the other hand, index of the original password of the user is maintained in "honeychecker" server. If we assume that for storing a single password, the existing system require (k-1)*n memory space. In this process, we assume that k is number of sweetwords and n is the number of users. For PDP method, the system requires (1+RS)*n storage space. RS is the random string in PDP methods [9].

Our proposed method takes (1*n) memory space and it can save storage overhead which is huge benefits.

TABLE I. COMPARATIVE STUDY OF HONEYWORDS GENERATION METHODS

No	Methods	Flatness	Typo Safety	Storage Overhead
1	CTD	1/k	Low	(k-1)*n
2	Password Model	1/k	High	(k-1)*n
3	Take a Tail	1/k	High	(k-1)*n
4	PDP	1/k	High	(1+RS)*n
5	Our Proposed Method	1/k	High	1*n

CTD- Chaffing by Tweaking
PDP- Paired Distance Protocol
k- Number of sweetwords in password file
n- Number of users
RS- Random String

VI. EXPERIMENTAL RESULT FOR HASHING ALGORITHM

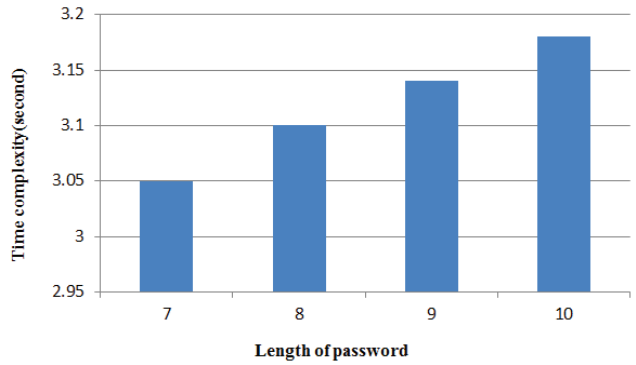
This password hashing scheme is implemented with python programming language**.** This programming language was

designed to meet all the real world requirements with its key features and easy for the programmers to learn and use efficiently. In this technique the most important aspect is less time complexity. The time complexity of an algorithm determines the amount of time taken by an algorithm to run as a function of the length of the string representing the input.

Suppose the user gives a password of different lengths such as 'hi45678, hello123, oranges321 and chocolate1234', etc. The time complexity of different length password in this scheme is in table II.

TABLE II
Time Complexity for length of password

Length of password (in characters)	Time complexity (second)
7	3.050004
8	3.100004
9	3.140004
10	3.18005



VII. CONCLUSION

Honey encryption technique using honeywords becomes various interesting challenging technique in security area because it can provide several advantages over password based scheme. In this paper, we present a novel honeywords generation method which has much lesser storage space and it can also reduce the majority of the drawbacks of the existing honeywords generation techniques. The honeywords generation method which is proposed here is an effective technique and can be used in honey encryption and decryption process. The proposed hashing and salting technique has much lesser time complexity than Advanced Encryption Standard (AES), Data Encryption Standard (DES) or any other contemporary technique. In the future, we will apply this honeywords generation method and hashing and salting algorithm in real world application system that is need for security such as message transmission process.

References

[1] Mirante, D and Justin,C, "Understanding Password Database Compromise", Technical Report TR-CSE-2013-02, Department of Computer Science and Engineering Polytechnici Institute of NYU, 2013.

[2] Vence, A, "If your password is 123456, just make it hackme", The New York Times 20, 2010.

[3] Brown,K , "The danger of weak hashes", Technical report, SANS Institute InfoSec Reading Room, 2013.

[4] Prof. Rohini S. More, Prof. Smita S. Konda, "Resilient security against hackers using enchanced encryption techniques: Blowfish and Honey Encryption" International Journal on Recent and Innovation Trends in Computing and Communication, Volume: 4 Issue: 6, June 2016.

[5] Ziya Alper Genc, Suleyman Kardas, Mehmet Sabir Kiraz, "Examination of a New Defence Mechanism: Honeywords," Inernational Journal of Engineering Trends and Technology (IJETT), Volume 27 Number 4, September 2015.

[6] Ari Jules, Ronald L. Rivest, " Honeywords: Making Password-e Cracking Detectable" MIT CSAIL, May 2, 2013.

[7] R. Gennaro and Y. Lindell, "A framework for password-based authenticated key exchange," In *Advances in CryptologyEUROCRYPT 2003*, pages 524–543. Springer, 2003

[8] Nirvan Tyagi [ntyagi], Jessica Wang [jzwang], Kevin Wen [kevinwen] and Daniel Zuo [dzuo],, "Honey encryption Application," Computer and network Security, Springer, 2015.

[9] Defense Information Systems Agency (DISA) for the Department of Defense (DOD), "Application security and development", Security technical implementation guide (STIG), version 3 release 4, 28 October 2011.

[10] S.Schechter, C. Herley and M.Mitzenmacher, "Popularity is everything: a new approach to protecting passwords from statical guressing attacks", USENIX HotSec, pages1, 2010.

[11] Nilesh Charkraborty and Samrat Mondal, "A New Storage Optimized Honeyword Generation Approach for Enhancing Security and Usability," 21, SEPT, 2015.