International Journal of Computer (IJC)
ISSN 2307-4523 (Print & Online)
© Global Society of Scientific Research and Researchers
http://ijcjournal.org/

Selecting the Honeywords from Existing User's Passwords Using Improved Hashing and Salting Algorithm

Khin Su Myat Moe[a]*, Thanda Win[b]

[a,b]*Department of Computer Engineering and Information Technology, Yangon Technological University, Yangon, Myanmar*
[a]*Email: sumyat3040@gmail.com*
[b]*Email: thanda80@gmail.com*

Abstract

Nowadays, hashing passwords become the most essential tool for various web applications for making login process. However, password hashing takes many times for processing and it has become easier for attackers to crack hashing passwords from legitimate users by using brute force attack. Brute force attack is one of the dangerous attacks for password hashing techniques. Therefore, the legitimate user accounts are stored the passwords with honeywords using honeywords generation algorithm in order to prevent from brute force attack. Honeywords generation method is to produce the fake or decoy password for deceiving the attackers. However, the existing honeywords generation algorithm meets the storage overhead problem. So, we are implementing the improved honeywords generation method which decreases the storage overhead problem and also it addresses the majority of the drawbacks of existing honeywords generation methods. 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: brute force attack; hashing passwords; honeywords generation algorithm; storage overhead problem; time complexity.

1. Introduction

The storage of passwords in the password file is one of the challenge problems in many application areas because many web applications use password for performing login process. So, the password files play an important role in millions of users and companies such as Yahoo, RockYou, LinkedIn, eHarmony and Adobe [5,11] since leaked password makes the user target of many possible cyber-attacks.

----

* Corresponding author.

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 [2].

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. The way they know they have the correct password is when the database or file becomes readable. To speed up the process, hackers 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 honey encryption (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 [8]. 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.

There have been several high publicity password leaks over the past year including LinkedIn, Yahoo, and eHarmony. While you never want to have vulnerabilities that allow hackers to get access to your password hashes, you also want to make sure that if the hashes are compromised it is not easy for hackers to generate passwords from the hashes [4]. But the existing hashing and salting algorithm take many time to store hashing passwords in the database.

Ziya Alper Genc, Suleyman Kardas, Mehmet Sabir Kiraz discussed that password is easy to crack with the improvement 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 [1].

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 problems. 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 [9].

Many researchers consider 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 appeared 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 [6].

Many password systems, particularly for government and industry users, store hashes of users' old passwords usually the last 10, as stipulated in, e.g., [3]. 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 not to store old passwords on a per-user basis.

The example of existing system architecture design of honeywords generation algorithm is shown in following fig. 1. In this figure, the user makes the registration process and the system generate honeywords for this user if the user is new user. The user’s real password and generating honeywords are converted into hexadecimal form using hashing and salting algorithm and stored into the database. If the user is member of this system, the user can login with his username and password. In this section, the server checks the entire password exists in the database. When the password exists in the database, the server sends this password to honeychecker for classification of honeyword and real password. Otherwise, the system sends unsuccessful login to user and quit the system. By checking the password from honeychecker, the system sends an alarm message to the system administrator if the password is honeyword. On the other hand, the user can enter the system successfully [12]. However, there are some limitations in honeywords generation process such as storage overhead problem, typo safety problem and so on.

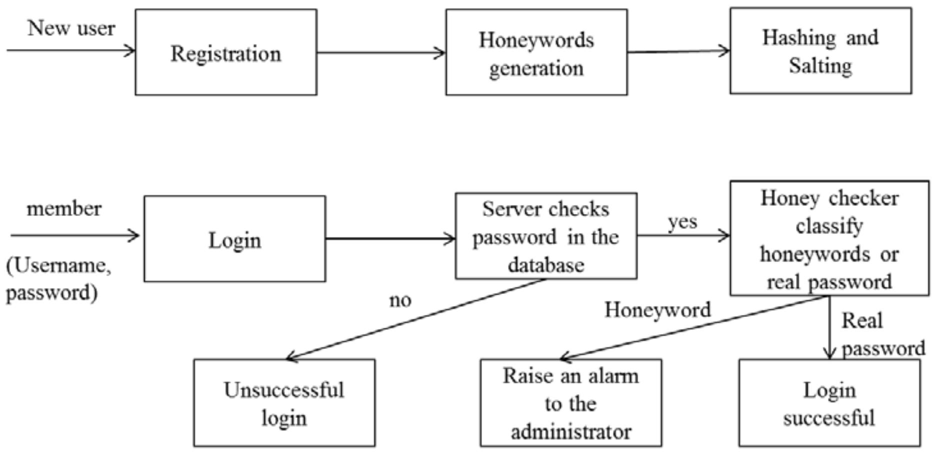

Figure 1: Existing System Architecture for Honeywords Generation

2. Background Theory

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

2.1. 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 [7]. 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.

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 safety store secret information and can raise an alarm to the administrator when an irregularity is detected. Table I has two attributes: first attribute 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 entered position. There are also two parts 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 1: Example of password file 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 2: Example of password file F2 in honey checker

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

2.2. Hashing and Salting Algorithm

In our 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 0's and 1's Step
Step 5: Rotate the string left or right by r character depending on the system.

Step 6: Convert the interchange binary string into hexadecimal string.

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 (yellow color) are added to the left side of 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 0's and 1's. In this case, the value of X is 4. It means String2 is formed with 4 zer0'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 operation, the resulting binary string are described as follows.

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.

3. System Flow of Our System

The flowchart for our system is shown in fig. 2. 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 or she tries to enter into system, the server checks whether this user's password exist or not into the database. If the password doesn't exist in the database, the server returns the unsuccessful login message to the user. If the 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.

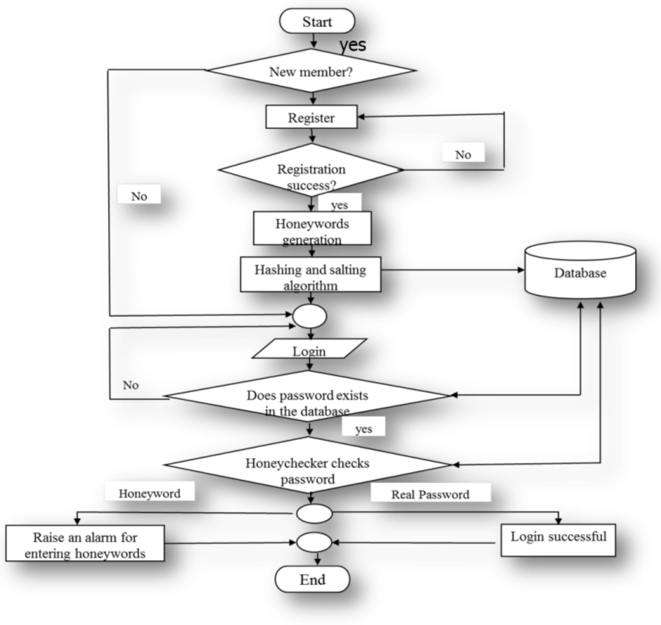

Figure2: Flowchart of the system

4. Comparative study on honeywords generation methods

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

4.1. Flatness

If the system maintains k sweetwords against a user ui then attacker may confuse among k possible options once list of Wi is compromised. The list of Wi contains one sugarword (the real password) and (k-1) honeywords .Now sometimes it may happen that t adversary can easily identify t password chosen by the user from the list Wi (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 Wi. If the honeyword generation algorithm is perfectly-flat then the probability of selecting the original password of user from list Wi is 1/k. If the probability of selecting user password from the list Wi 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 [3].

Our honeywords generation method is a good honeywords generation method and the probability of selecting the original password of user from password file is 1/k. So, our system is perfectly flat because we use the other user’s passwords as honeywords for reducing storage overhead problem.

4.2. 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 sweetwords [3]. 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.

4.3. Storage Overhead

Using the existing honeyword generation algorithms system maintains k-1 extra passwords along with the riginal 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. [3] Our method takes (1*n) memory space and it can save storage overhead which is huge benefits.

The comparison analysis of existing honeywords generation methods and our method is shown in the following table 3.

Table 3: Comparative study of honeyword generation method

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 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

5. Experimental result of hashing algorithm

This password hashing scheme is implemented in python**.** 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. Unlike other programming system, python provides a small number of clear ways to achieve a given task. 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 length like 'hi45678, hello123, oranges321 and chocolate1234'. Then the time complexity of different length password in this scheme is as follows:

Table 4: Time complexity for length of password

Length of password (characters)	Time complexity (second)
7	3.050004
8	3.060004
9	3.130004
10	3.250005

6. Security Analysis of the System

Password based encryption (PBE) algorithm plays an important role in various areas such as business groups, banking system and computer users. But PBE is not much safe to provide the security to the users because large number of attacks. Here it examines the security of the our system against some possible attack which is follows.

6.1. Brute Force Attack

Brute force attack is method used by attackers to decode encrypted data such as passwords or data encryption standard key. This attack consists of systematically checking all possible combination of password or keys. But in this encryption technique, this attack is not possible due to the honeywords generation methods. Brute force attacks are very time consuming because of hashing algorithm. Moreover, the attackers cannot classify which is real key or password if they get password file.

6.2. DoS Attack

Denial of service attack is a kind of attack to make a machine or network resources unavailable to its users. The purpose of this proposed design is to minimize the DoS vulnerabilities. When a person tries to login into the system, the system automatically check login of the user to detect Dos attack. If the attacker tries to login with his attacking password file, the system raises alarm for user and attack detected. And then, the system can block that particular user.

6.3. Password Guessing Attack

In this attack, the attacker is to steal password file F1 and F2 from the main server and honey checker. And then, he tries to get original password by reversing the hash value. Firstly, he take the F2 file < index no, password> pair but he cannot know which user password belongs to which user account because it is not directly connected to a specific username or account. Otherwise, if the attacker attacks F1 password file <username, honeyindex set> pair, he has no advantages in guessing correct password due to the honeyindex. When the attacker randomly chooses an account and then tries to login with guessed password from password file, it can be two probabilities which are successful or fail. First is taking the password from the password file is correct and the next is getting the honeywords. If the attacker will login with honeywords, the system raises an alarm for the user account and can detect this attack.

7. Conclusions

Password security has always been an interesting challenges technique in security of various areas using passwords. Honeywords based authentication can provide several advantages over password based system. The big difference between the traditional password based system and when honeywords are used is that a successful brute forces attack does not gives the attacker confidence that can log in into system successfully without being detected. In this study, 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.

Moreover, our method can be used as an effective method in honey encryption and decryption process. The simple hashing and salting algorithm needs the few seconds for storing the passwords as hexadecimal form into the database. In the future, we will apply this honeywords generation method in real world application system that is needed for security such as message transmission process in military, government offices and so on.

References

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

[2] Brown,K . “The danger of weak hashes,” Technical report. SANS Institute InfoSec Reading Room. 2013.

[3] 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 Oct. 2011.

[4] K. Brown, “The Dangers of Weak Hashes,” SANS Institute InfoSec Reading Room, Tech. Rep., 2013

[5] 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.

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

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

[8] 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, vol. 4, Issue: 6, June. 2016.

[9] R. Gennaro and Y. Lindell. “A framework for password-based authenticated key exchange,” In Advances in CryptologyEUROCRYPT 2003, Springer, 2003, pp 524–543.

[10] S.Schechter, C. Herley and M.Mitzenmacher. “Popularity is everything: a new approach to protecting passwords from statical guressing attacks,” USENIX HotSec, 2010, pp1.

[11] Vence, A. “If your password is 123456, just make it hackme”, The New York Times (2010).

[12] Ziya Alper Genc, Suleyman Kardas, Mehmet Sabir Kiraz. “Examination of a New Defence Mechanism: Honeywords,” Inernational Journal of Engineering Trends and Technology (IJETT), vol. 27, Number 4, Sept. 2015.