Analysis of Password and Salt Combination Scheme To Improve Hash Algorithm Security

Sutriman[1], Bambang Sugiantoro[2]
Master of Informatics Department
Sunan Kalijaga Islamic State University
Yogyakarta, Indonesia

***Abstract*—In system security, hashes play important role in ensuring data. It remains the secure and the management of access rights by those entitled to. The increasing power of hash algorithms, various methods, are carried out one of them using salting techniques. Salt is usually attached as a prefix or postfix to the plaintext before hashing. But applying salt as a prefix or postfix is not enough. There are so many ways to find the plaintext from the resulting cipher text. This research discusses the combination scheme other than the prefix and postfix between password and salt increasing the security of hash algorithms. There is no truly secure system and no algorithm that has no loopholes. But this technique is to strengthen the security of the algorithm. So that, it gives more time if an attacker wants to break into the system. To measure the strength generated from each combination scheme, a tool called Hashcat is used. That is the way known as the best composition in applying salt to passwords.**

Keywords—Security; hash; hashing scheme; salting; password

I. INTRODUCTION

Hash is an algorithm that changes the string becomes a series of random characters. It is also called a one-way function, or one-way encryption because it is only able to do encryption and does not have a key to decrypt. It works by accepting input strings that are arbitrary in length then transform it in a string of fixed length which is called hash value [1][2][3][4].

Hash is often used to provide security to the authentication process. An authentication is a process of ensuring a property is genuine, verifiable and trustworthy; deep conviction the validity of the transmission, message, or sender of the message. It verifies that the user should input entered from the system coming from a trusted source [1].

Authentication is one of several concepts needed to ensure the security of a system. Authentication along the accountability is the additional concept needed to support the CIA Triad. CIA Triad is a concept very well-known as the security, named the Confidentiality, Integrity, and Availability [1][5]. CIA triad is the basic model of Information Security and there exist other models that have the attributes of the CIA triad in common [6]. Despite the use of the CIA to determine goals security is well established.A few in the security sector feels that the additional concept is needed to present the picture completely [1].

Authentication is a very important process because besides maintaining information from unauthorized users. It also maintains the integrity data [7][8]. The use of algorithms and hashing techniques is needed to help the authentication process so that they can minimize the occurrence of broken data by the attacker. The authentication process utilizes the use of algorithms hash including the authentication of login (password), authentication file authenticity, password storage, key generation, pseudorandom number generation, authentication of tokens on services in a distributed system, digital signature, etc. [9].

In information systems, a hash is used for the authentication login process. Passwords are changed using certain hashing methods thus producing unique characters later stored in the database. Some common hash functions used include MD5 and SHA1 [3]. A message digest (MD) is the code which is created algorithmically from the file and represents that file uniquely. If the file changed, the message digest will change [10]. Message Digest describes the mathematical function that can take place on a variable-length string. The number five (5) simply depicts that MD5 was the successor of MD4. MD5 is essentially a checksum that is used to validate the authenticity of a file or a string. It is one of the most common uses [11]. The MD5 algorithm exhibits a lot of weaknesses such as its vulnerabilities to different attacks such as rainbow table, dictionary, birthday, etc. [12]. SHA is a series of cryptographic hash functions designed by the National Security Agency (NSA). The weakness in SHA family originated from this fact that possibility of two different input value will produce the same output value in the middle of algorithm and it is important to have a good diffusion. So, the output in each round will be spreaded out and not to be equal with the same output in the next coming stages [13]. MD5 and SHA1 are hash algorithm that do not recommend. MD5 and SHA1 have many vulnerabilities which allow attackers to easily get the system user password by knowing the hash value.

A new hash algorithm appears as time progresses with better security than the previous algorithm, among them are SHA2, SHA3, BCrypt, and others. Along with the development of the era, no doubt the new algorithms even the vulnerability of attackers will be found.

The use of hashes in the authentication process actually can reinforce by adding salt to the plaintext password before the hashing process is carried out. Salt in the cryptography is a

random bit that is used as a joint input password before the hashing process is done. Salt can be added to the hash to prevent a collision by uniquely identifying a user's password, even if another user in the system has selected the same password [14]. Salt is used as prefixed or post fixed against passwords before entering the process hashing. Each password has different salt.

The use of salt for passwords can increase password security in an application, but does not close possibilities that the attacker can crack against the generated values. Various kinds of password cracking tools who are currently circulating in cyberspace started by using password and salt combination for plaintext password guessing. Giving salt a prefix or postfix still has a vulnerability.

In [15] is previous research that has been conducted that discusses the method of exchanging passwords and salt. In that study, each index in the password and salt was exchanged and then stored in an array. Unfortunately, the research does not show the results of strength measurements that could assess the results of the study clearly. So, raising the question is a really measured thing. Then it can be added from the other parameters as a comparison.

II. Related Work

Password is one of the most important components of any classical security scheme designed to protect sensitive and confidential information from falling into the wrong person. Creating a strong password is the first step towards ensuring the protection of confidential user information [16]. The purpose is to improve password security and to contribute the research on password security, including salting techniques. Salting technique is a hedge against pre-computed dictionary attacks, the bedrock of which involves concatenating a random string of letters and numbers, a salt, to the beginning or end of a password before hashing it [17].

Abdelrahman Karrar *et al.* have published research on the security of hash algorithms by swapping every index for passwords and salts. Swapping Elements in an array algorithm consists of two main modules, the Hash input structuring module and the Salt rearrangement module[18].

The previous research on the use of salt strengthen hashes on passwords has also been carried out by combining with differential masking techniques [15]. The differential masking is basically the insertion of fake passwords associated with each user's account. When an attacker gets the password list, he retrieves many password candidates for each account but cannot be sure about which password is real[19]. The use of various methods and combinations is needed to increase the power of hash algorithms. However, only using salt is not enough because it can still be attacked using the Bruteforce attack technique for only a little longer. Using two salts, one public and one private can also protect the password against offline attacks [20].

III. Research Method

Research methods are a way to collect various data processed into information. The information is used as materials to solve the problems studied. The steps taken in this study include data collection, data grouping, analysis process and ending with conclusions obtained. The research method used in this study is shown in Fig. 1.

A. Research Tools

Research tools are used to carry out the research process. In the testing process, the specifications of the research tool affect the test results. The research tool used in this study is a computer laptop with specifications as following:

- Brand/Type : HP / G 240 G7
- Processor : Intel Core i7-8565U
- RAM : 8 GB DDR 4
- Storage : 256 GB SSD
- Operating System : Linux, Xubuntu 18.04

B. Generating Salt Process

In this process, the salt which will be combined with the password is made. The salt is made using a random function that is owned by the PHP programming language. It plays an important role in making hashed passwords stronger. By using the salt the more unique and the longer, a simple password will become stronger. In this study, the salt generated is limited to only 3 character numbers to simplify the testing process.

C. Rearrangement Process

This process is a process that will be carried out to strong the hash algorithm. The use of salt is generally only combined as a prefix or postfix for passwords. In other cases, the use of salt composition is very dependent on the programmer who built a system. By doing randomization between passwords and salt, it is expected that a plaintext will be carried out by the hashing process. It will be more difficult to describe by the attacker, but without adding too many characters, it does not increase the hashing time to be longer.

D. Testing

The tests that will be carried out in this study use a process that can find the plaintext of the password. The treatment for each scheme will be different based on each combination. Each combination will be described, and the time needed to be able to decipher the ciphertext of the saved password recorded.

The decomposition process, at this stage of testing, will use a simple algorithm made in Python language while for the hash parsing process a password recovery tool will be used, named OclHashcat. It is the fastest password recovery program based on GPU, built by atom and can run on GNU or Linux and MS Windows, 32 and 64 bit [21].

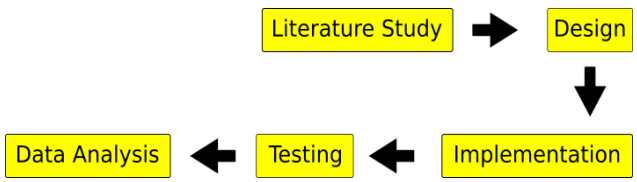

Fig. 1. Research Methods.

IV. Results and Discussions

A. Designing a Password and Salt Combination Scheme

Some draft password and salt combination schemes are needed to be able to compare the strengths of each design. The scheme created is a combination of several in the hash process between passwords and salt. The scheme used in this study can be seen in Table I:

TABLE. I. Password and Salt Combination Scheme

No	Scheme	Combination
1	Scheme 1	hash(rearrangement(password, salt))
2	Scheme 2	hash(rearrangement(password, salt)+salt)
3	Scheme 3	hash(hash(rearrangement(password, salt)))

Scheme 1 is the same scheme used by previous researchers. Schemes 2 and 3 are schemes created as a comparison by providing additional parameters. In scheme 2, the salt parameter is added to the exchange result before the hash process is performed. In scheme 3, a double hash is performed on the result of changing the password and salt position.

B. Implementation

Implementation is the process of implementing a scheme that has been designed into an application. The application made in this section is a simple application that can display the cipher text value of each designed combination scheme.

The implementation is done using the PHP and MySQL database. The use of databases in system implementation aims to get records of each result that is generated during the process of collecting data with the application that has been made.

1) Generating Salt: In this process, the salt which will be combined with the password is made. Salt is made using a random function that is owned by the PHP programming language. Salt plays an important role in making hashed passwords stronger. By using salt the more unique and the longer, a simple password will become stronger.

In this study, the salt was generated using the mt_rand() function that is already available in the PHP programming language and limited to only 3 character numbers to simplify the testing process. The mr_rand() function is used to generate a random number value between the given range. The function used to generate salt in its entirety can be seen in the program code below:

```
function randomSalt($length) {
        $result = '';
        for($i = 0; $i < $length; $i++) {
                $result .= mt_rand(0, 9);
        }
        return $result;
}
```

Script. 1. Generating Salt Function

2) Rearrangement Process: This process will be carried out to strengthen the hash algorithm. The use of salt is generally combined as a prefix or postfix for passwords. In other case, the use of salt composition is very dependent on the programmer built a system. By doing randomization between passwords and salt, it will be carried out by the hashing process will be more difficult to describe by the attacker without adding too many characters so that it does not increase the hashing time to be longer.

```
function rearrange ($arr1, $arr2, $n1, $n2){
        $i = 0;
        $j = 0;
        $k = 0;
        $arr3 = array();
        while ($i < $n1 && $j < $n2){
                $arr3[$k++] = $arr1[$i++];
                $arr3[$k++] = $arr2[$j++];
        }
        while ($i < $n1)
        $arr3[$k++] = $arr1[$i++];
        while($j < $n2)
        $arr3[$k++] = $arr2[$j++];
        $result = array();
        for ($i = 0; $i < ($n1 + $n2); $i++)
        array_push($result, $arr3[$i]);
        return implode("", $result);}
```

Script. 2. Rearrangement Function

The rearrangement() function above is called by including 4 parameters. The first and second parameters, namely $arr1 and $arr2 are passwords and salts. The third and fourth parameters namely $n1 and $n2 are the length of the password and salt. The above function works by rearranging the password and salt. For example, the password that is owned is abcdef and salt 123. It will produce an a1b2c3def string.

3) Data Collection: Data retrieval is the process of recording each cipher text value of each combination scheme for each predetermined password. The password used is a weak password taken from Splash Data's Top 100 Worst Passwords. From the collection of passwords found. The ones used in this study are those that match the criteria which are 6 characters long. The passwords used can be seen in Table II.

TABLE. II. Sample Password

No	Password
1	qwerty
2	monkey
3	abc123
4	123123
5	dragon
6	qazwsx
7	654321
8	harley

C. Testing

In this test, it is assumed that the examiner who acts as an attacker knows the combination scheme model and length and the type of data used for passwords and salts used as tested material. This aims are to limit research because the attacking process is very dependent on information and analysis obtained by the attacker. Thus, what was tested in this study was pure to measure the effect of the combination scheme on the strength of the hash algorithm.

1) Testing using Hashcat: Hash cat is a penetration testing tool that can be used to decrypt cipher text results from the hashing process. Hash cat has variety of attack methods with various hash decrypted models. In the testing phase of this study, what will be used is the Mask Attack model.

Mask Attack is an attack mode that uses a combination of characters to guess the plaintext sought. The Mask Attack on the Hashcat is similar to Brute force Attack. In traditional brute force mode, a character set is needed that contains all uppercase letters, all lowercase letters and all digits (mix alpha-numeric). So, it takes a long time.

In a Mask Attack mode, character sets can be arranged based on information about the target that has been obtained. In this test, not all mix alpha-numeric characters are used to carry out attacks but can be adjusted to the desired attack pattern based on the information that has been obtained. One example of the command used on Attack Mask Attack using Hashcat is as follows:

hashcat -a 3 -m 0 scheme3_monkey.txt ?l?d?l?d?l?d?l?l?l --force

For,

hashcat : Unique key used to call applications

-a : Attack Mode (3 to define the Mask Attack mode)

-m : Hash Type (0 to define that the target hash is MD5)

scheme3_monkey.txt : Chipertext as the target

?l?d?l?d?l?d?l?l?l : Characters used to determine the attack pattern,? l for letters a-z,? d for numbers 0-9.

2) Reverse Rearrangement Process: This stage is used because schemes that uses the rearrangement function before the plaintext is converted to the Ciphertext in the hashing process need to be rearrange. At this stage, the special scripts are created with python language which can be seen in the following script:

```
def main():
  input_string = raw_input("Input your string: ")

  start_time = time.time()
  start_time_print = time.strftime('%d/%m/%Y %H:%M:%S')
  plaintext_length = 6

  salt_length = 3
  string_to_list = list(input_string)

  list_of_result = []
  for x in xrange(0,len(input_string)):
    new_index = x
    if x != 1 and x != 3 and x != 5:
      list_of_result.insert(new_index,string_to_list[x])
  result = ''.join(list_of_result)
  print result
  print("\n\n\n--- start time: %s" % start_time_print)
  print("--- stop time: %s" % time.strftime('%d/%m/%Y %H:%M:%S'))
  print("--- time estimated %s seconds" % (time.time() - start_time))
```

Script. 3. Script for Reverse Rearrangement.

In the above function, the input string is a password and salt combination string. This function works by looping with the password length and salt parameters that were previously known. In the looping process, characters identified as part of the password stored in the new list and displayed as a result.

3) Test Results: The main purpose of the testing process is the time needed to get the plaintext for each scheme. The results of this test will then be analyzed in the next step. The test results are shown in the following Tables III, IV and V.

TABLE. III. TEST RESULT FOR SCHEME 1

No	Password	Salt	Time		
			Cracking Hash	***Reverse Rearrangement***	***Total***
1	qwerty	857	1h7m57s	0.000262975692749s	1h7m57.0003s
2	monkey	047	1h4m57s	0.00019907951355s	1h4m57.0003s
3	abc123	565	7s	0.000290155410767s	7.0003s
4	123123	345	5s	0.000843048095703s	5.0008s
5	dragon	372	1m16s	0.000288963317871s	1m16.0003s
6	qazwsx	512	3h35m17s	0.000334024429321s	3h35m17.0003s
7	654321	782	8s	0.000181913375854s	8.0002s
8	harley	456	1h6m14s	0.000261068344116s	1h6m14.0003s

TABLE. IV. TEST RESULT FOR SCHEME 2

No	Password	Salt	Time		
			Cracking Hash	*Reverse Rearrangement*	*Total*
1	qwerty	857	2h12m36s	0.000263929367065s	2h12m36.0003s
2	monkey	047	2h9m1s	0.000308990478516s	2h9m1.0003s
3	abc123	565	17s	0.000260829925537s	17.0003S
4	123123	345	16s	0.000319004058838s	16.0003s
5	dragon	372	12m33s	0.000279903411865s	12m33.0003s
6	qazwsx	512	4h18m40s	0.000226020812988s	4h18m40.0002s
7	654321	782	35s	0.000211000442505s	35.0002s
8	harley	456	1h50m22s	0.0001380443573s	1h50m22.0001s

TABLE. V. TEST RESULT FOR SCHEME 3

No	Password	Salt	Time		
			Cracking Hash	*Reverse Rearrangement*	*Total*
1	qwerty	857	1h48m39s	0.000224113464355s	1h48m39.0002s
2	monkey	047	1h57m11s	0.000231027603149s	1h57m11.0002s
3	abc123	565	18s	0.000243902206421s	18.0002s
4	123123	345	16s	0.000253200531006s	16.0002s
5	dragon	372	12m18s	0.000326156616211s	12m18.0003s
6	qazwsx	512	4h8m59s	0.000308990478516s	4h8m59.0003s
7	654321	782	29s	0.000277996063232s	29.0003s
8	harley	456	1h49m40s	0.000302791595459s	1h49m40.0003s

D. Data Analysis

At this step, the test results will be processed to be more informative. Data samples are eight (8) in total for each combination scheme. It was looked for an average value so that they can be easily compared with other combination schemes.

1) Look for the average value of each combination scheme: Before the process is done to find the average value, the total value of time which is still in the form of hours, minutes and seconds must be equalized in the form of seconds. The average is found by dividing the number of values from the entire data in one combination scheme, the amount of data, or the number of samples used. The formula used is as follows:

$$\bar{x} = \frac{\sum x}{N}$$

For,

$\bar{x}$ = Average value

$\sum x$ = Total value of x

N = the amount of data

The results of the average calculation for each scheme can be seen in Table VI.

2) Average Graph: In Table VI we can see the comparison of the average time needed to get the plaintext from each scheme. The average time indicates how strong the scheme has been designed. The higher required time can be interpreted the stronger the scheme.

The average time graph from Table 6 can be seen in Fig. 2 below:

TABLE. VI. THE AVERAGE ATTACK TIME FOR EACH SCHEME

No.	Scheme	Average Time (s)
1	1	3120.12
2	2	4832.50
3	3	4483.75

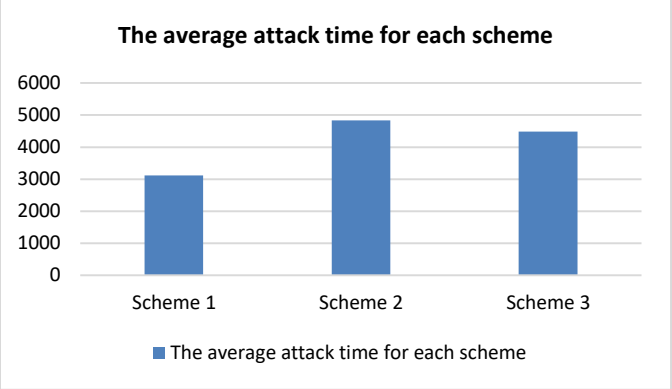

Fig. 2. The Average Attack Time for Each Scheme.

V. CONCLUSIONS

In this paper, we evaluate the strength of several combination schemes between passwords and salts. The scheme is tested based on the arrangement of the position exchange of each index between the password and the salt. With this research, it is hoped that knowledge can be obtained. The preparation and the addition of a few parameters can significantly increase the strength of the hash algorithm.

The hope for the future, there is further research on this field. For example, by adding another scheme, other parameters or with a truly random arrangement. Because this research is important in data security.

REFERENCES

[1] W. Stallings and L. Brown, Computer Security: Principles and Practice, Global Edition. 2015.

[2] N. Mouha, M. S. Raunak, D. Richard Kuhn, and R. Kacker, "Finding Bugs in Cryptographic Hash Function Implementations," IEEE Trans. Reliab., vol. 67, no. 3, pp. 870–884, 2018.

[3] P. P. Pittalia, "A Comparative Study of Hash Algorithms in Cryptography," vol. 8, no. 6, pp. 147–152, 2019.

[4] M. Cindy, A. Kioon, Z. Wang, and S. D. Das, "Security Analysis of MD5 Algorithm in Password Storage Security Analysis of MD5 algorithm in Password Storage," no. February 2013, 2015.

[5] S. Qadir and S. M. K. Quadri, "Information Availability: An Insight into the Most Important Attribute of Information Security," J. Inf. Secur., vol. 07, no. 03, pp. 185–194, 2016.

[6] J. Andress and S. Winterfeld, "The Basics of Information Security: Understanding the Fundamentals of InfoSec in Theory and Practice: Second Edition," pp. 1–217, 2014.

[7] P. Ramos Brandão, "The Importance of Authentication and Encryption in Cloud Computing Framework Security," Int. J. Data Sci. Technol., vol. 4, no. 1, p. 1, 2018.

[8] M. Trnka, T. Cerny, and N. Stickney, "Survey of Authentication and Authorization for the Internet of Things," Secur. Commun. Networks, 2018.

[9] A. Sahu, "Review Paper on Secure Hash Algorithm With Its Variants International Journal of Technical Innovation in Modern Engineering & Science (IJTIMES) Review Paper on Secure Hash Algorithm With Its Variants," no. May 2017, pp. 0–7, 2018.

[10] R. Mohanty, N. Sarangi, and S. K. Bishi, "A secured Cryptographic Hashing Algorithm," Analysis, p. 4, 2010.

[11] A. K. Kasgar, M. K. Dhariwal, N. Tantubay, and H. Malviya, "A Review Paper of Message Digest 5 (MD5)," Int. J. Mod. Eng. Manag. Res., vol. 1, no. 4, 2013.

[12] A. Bhandari, M. Bhuiyan, and P. W. C. Prasad, "Enhancement of MD5 Algorithm for Secured Web Development," J. Softw., vol. 12, no. 4, pp. 240–252, 2017.

[13] H. Mirvaziri, K. Jumari, M. Ismail, and Z. M. Hanapi, "A new hash function based on combination of existing digest algorithms," 2007 5th Student Conf. Res. Dev. SCORED, no. December, pp. 1–6, 2007.

[14] P. N. Patel, J. K. Patel, and P. V Virparia, "A Cryptography Application using Salt Hash Technique," Int. J. Appl. or Innov. Eng. Manag., vol. 2, no. 6, pp. 236–239, 2013.

[15] S. Kharod, N. Sharma, and A. Sharma, "An improved hashing based password security scheme using salting and differential masking," 2015 4th Int. Conf. Reliab. Infocom Technol. Optim. Trends Futur. Dir. ICRITO 2015, pp. 1–5, 2015.

[16] E. M. W. R. Chowdhury, M. S. Rahman, A. B. M. A. Al Islam, and M. S. Rahman, "Salty Secret: Let us secretly salt the secret," Proc. 2017 Int. Conf. Networking, Syst. Secur. NSysS 2017, pp. 115–123, 2017.

[17] J. Zhang and S. Boonkrong, "Dynamic salt generating scheme using seeds warehouse table coordinates," 2015 IEEE 2nd Int. Conf. InformationScience Secur. ICISS 2015, 2016.

[18] A. Karrar, T. Almutiri, S. Algrafi, N. Alalwi, and A. Alharbi, "Enhancing Salted Password Hashing Technique Using Swapping Elements in an Array Algorithm," vol. 8491, pp. 21–25, 2018.

[19] D. Mirante and J. Cappos, "Understanding Password Database Compromises Technical Report," Tech. Rep. TR-CSE-2013-02, Polytech. Inst. NYU, 2013.

[20] K. Chanda, "Password Security: An Analysis of Password Strengths and Vulnerabilities," Int. J. Comput. Netw. Inf. Secur., vol. 8, no. 7, pp. 23–30, 2016.

[21] Radix, "Hashcat User Manual," no. August, 2011.