The Efficiency of using Salt Against Password Attacking
ประสิทธิภาพของการใช้ซอลท์กับการโจมตีรหัสผ่าน

Wachana Khowfa
Onsiri Silasai
Faculty of Science and Technology, Suan Dusit University

Received: November 17, 2017
Revised: January 5, 2018
Accepted: January 8, 2018

วัจนา ขาวฟ้า*
อรศิริ ศิลาสัย
คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยสวนดุสิต
*e-mail: wachana_kho@dusit.ac.th

Abstract

Password is used in the process of authentication and gaining the access right to get into the system, therefore, password must be stored in secured place and away from any type of password attack. Hash function is used to protect plain text password whenever the password is attacked. However, hashed value of password generated from the faster hashing formulation can still be easy to break. Another technique to provide strength to password is adding a set of string called salt into password before performing hash function. The objectives of this study included 1) to enhance the security of weak password and 2) to evaluate if the position of placing salted value has significant to the strength of the password. This research started from selecting 10 weak passwords. The position of placing salted value consists of 1) prefix 2) suffix and 3) inserted salt which considered by the frequency alphabet used. When high rate of letter using is found, salt value will be placed after that particular letter. After the process of placing salt, all passwords are performed the password attacking via Dictionary Attack and Brute Force Attack. The result stated that the use of salt can significantly enhance the level of difficulty and complexity to crack password and can improve the attack resistance level of weak password to meet the same security level as strong password. Moreover, the position of salt insertion has notable significance to the security level of password which consists of frequency letter.

Keywords: Password Attacking, Password Attacking Resistance, Dictionary Attack, Brute Force Attack, Salting Password

บทคัดย่อ

รหัสผ่านถูกนำมาใช้ในกระบวนการยืนยันตัวตนและการได้รับสิทธิ์เข้าถึงการใช้งานระบบ ดังนั้น รหัสผ่านควรได้รับการจัดเก็บอย่างปลอดภัยและไม่สามารถโจมตีได้ง่ายจากการโจมตีรหัสผ่านในรูปแบบต่าง ๆ การทำแฮชฟังก์ชั่นถูกนำมาใช้ในการป้องกันรหัสผ่านที่ประกอบด้วยตัวอักษรทั่วไปที่อ่านเข้าใจได้ในกรณีที่เกิดการโจมตีรหัสผ่าน อย่างไรก็ตามค่าแฮชของรหัสผ่านที่ได้จากการทำแฮชฟังก์ชั่นแบบรวดเร็วนั้นก็ยังคงง่ายต่อการถูกโจมตี เทคนิควิธีอีกอย่างที่นำมาใช้ในการเพิ่มระดับความปลอดภัยให้กับรหัสผ่าน คือ การเพิ่มชุดของกลุ่มคำที่เรียกว่า ซอลท์ เข้าไปในรหัสผ่านก่อนที่จะทำแฮช วัตถุประสงค์ของการวิจัยนี้ประกอบด้วย 1) เพื่อเพิ่มความปลอดภัยให้กับรหัสผ่าน และ 2) เพื่อประเมินตำแหน่งในการวางค่าซอลท์มีความสำคัญต่อความปลอดภัยของรหัสผ่านหรือไม่ ขั้นตอนการดำเนินการวิจัยเริ่มจากการเลือกรหัสผ่านที่ไม่มีความปลอดภัย จำนวน 10 ตัว ตำแหน่งในการวางของค่าซอลท์ที่ใช้ในการทดลองครั้งนี้ประกอบด้วย 1) การวางค่าซอลท์ไว้ข้างหน้ารหัสผ่าน 2) การวางค่าซอลท์ต่อท้ายรหัสผ่าน และ 3) การแทรกค่าซอลท์เข้าไปในรหัสผ่าน โดยพิจารณาจากระดับความถี่ในการใช้ตัวอักษรแต่ละตัว หากมีความถี่ในการใช้มากจะเพิ่มค่าซอลท์ต่อท้ายอักษรนั้น จากนั้นจึงทดสอบการโจมตีรหัสผ่านด้วยวิธีการแบบพจนานุกรมและการโจมตีแบบบรู๊ทฟอร์ช ผลการทดสอบ พบว่า การใช้ซอลท์ทำให้รหัสผ่านที่ไม่ปลอดภัยนั้นยากและซับซ้อนต่อการโจมตีมากยิ่งขึ้น และยังสามารถเพิ่มระดับความทนทานต่อการโจมตีรหัสผ่านที่ไม่ปลอดภัยให้อยู่ในระดับเดียวกับรหัสผ่านที่มีความปลอดภัยอีกด้วย นอกจากนี้ ตำแหน่งในการวางค่าซอลท์มีความสำคัญอย่างยิ่งต่อความปลอดภัยของรหัสผ่านที่ประกอบด้วยตัวอักษรที่มีการใช้งานบ่อย

คำสำคัญ: การโจมตีรหัสผ่าน ความทนทานต่อการโจมตีของรหัสผ่าน การโจมตีแบบพจนานุกรม การโจมตีแบบบรู๊ทฟอร์ช การซอลท์รหัสผ่าน

Introduction

The use of password provides users an authentication to prove their identity moreover to gain the access right into the system. As each password belongs to a single user, the password must be kept into secured location or the password must be strong enough to handle with any type of attack from attacker. In fact, many users decide to use same password to verify themselves and gain the access right into every system to reduce their memory load. Words existing in dictionary and sequenced letter on computer keyboard are commonly selected by users as their password. At the same time, sequencing number and set of numbers that related to user such as birth date and mobile phone number are always used as password as well. Thus, it is easy for attacker to create password database used to hack the password. When hacker obtains the password to get into one system, another system of particular user can be easily accessed into as well. Presently, the problems of password attack are available for example Dictionary Attack, Brute Force Attack and Rainbow Table Attack. The implementation of hash function techniques such as MD5, SHA1 and SHA 256 are applied in order to provide the security from password attack. However, with fast operating of hashing formula, the password can rapidly and easily to crack Provos & Mazières (1999 as cited in SHA, 2016). There are some solutions come over this problem such as performing slow hash function algorithm instead or applying extra word called salted value into password. In this paper Dictionary Attack and Brute Force Attack will be demonstrated. Dictionary Attack is the most commonly method in password attack by matching the password with word exist in dictionary. Thus, when user select the word from dictionary to use as their password, it can be easily to break. While Brute Force Attack is a time-consuming technique however is the successfully password attacking method by searching every single character in password until the password is cracked. The problem of password cracking will be discussed follow by the background of password storing,

password cracking and salting in section 2. A review literature of related works will be presented in section 3. In section 4, the experimental of cracking ten weak passwords in four formats h (original password), h (salt|| password), h (password||salt) and h (inserted salt) will be performed follows by the discussion on testing result. A conclusion and idea for future work will be stated in section 5.

Background Knowledge

A. Password Storing

Password plays an important role in user authentication process as it verifies the right of user in their system. Typical way to store user's password is to keep it into password database server without any protection as shown in Figure 1 in addition. The password that stored in clear text is easy to steal and misused by hackers due to the available type of cyber-attack. For this reason, password needs to be securely kept.

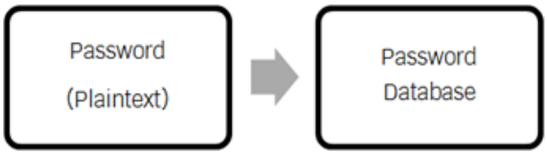

Figure 1 Clear Text Password Storing

Hash function is used to generate a digest of password before kept into password database instead of the clear password. Main reason is to protect password from an attacker (Stallings, 2014). Password in plaintext will be operated by the hash function as a result h (password) is created then stored in password database as shown in Figure 2. A hashed format of password similarly used as digital fingerprint of the original password to construct more security to password (Provos & Mazières, 1999).

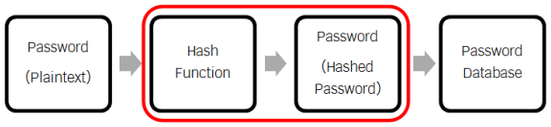

Figure 2 Hashed Password Storing

Presently, there are several different ways to calculate hash value. Secure Hash Algorithm (SHA) is one of the most common methods used in the process of password hashing. SHA was introduced in 1993 by National Institutes of Standards and Technology (NIST) and NSA (National Security Agency) (SHA256, 2016). The well-known SHA family is SHA1 and SHA2. As SHA is based on MD5 thus the step to perform hash computation is similar to MD5. However, SHA differs from MD5 by extending the block size of password used in calculation process moreover increasing operating round (SHA, 2016). SHA1, also designed by Ronald L Rivest, was found weakness in 1994. Therefore, SHA2 is assigned to use instead. SHA2 family consists of SHA224, SHA256, SHA384 and SHA512. The set of digit number after word "SHA" represents number of bit digest message (Raheja, Verma & Raheja, 2014). SHA2 family is presently a strong hash function by generating a longer message than SHA1 thus the suitable and generally used to protect the password.

B. Password Cracking

Password cracking is the process of trying to guess or matching user's password with the key word from attacker's database in order to gain the access right into authorized system (Chester, 2015; Tasevski, n.d.). However, cracking the password can be used for some other reasons such as recovering the forgotten password for user or performing the preventive measurement by system administrator to confirm the strength of password as well. Password cracking can be performed both online via the internet and offline using password cracking

applications. There are many methods of password attacking available at present however the common techniques that gain successfully of attacking are as follow:

1. Dictionary attack is a method of trying to guess password by matching the h (password) with a list of hashed word from a dictionary. If the password that user selected is a word existing in dictionary, then it can be found and broken very quickly (Kioon, Wand & Das, 2013 Tasevski). Typical way that password cracking works is to get a file containing user's password and then runs cracker software against the file to match all with of the rest word (Chester, 2015; Kioon, et al., 2013; Majumder, 2012). Important key successes of running dictionary attack are the input dictionary word selected and the word colander rules used. Colander rule is state that used as a filter or an extra condition to match word in dictionary and password together (Tasevski). The example of colander rule is adding capitalization the first letter, inserting 4 digits to the end of word or changing the letter from "i" to "1".

2. Brute force attack is known as a time-consuming method to crack password (Raza, Iqbal, Sharif & Haider, n.d.). To break password which consists of 8 lower alphabets, the hacker has to try 268 times which takes approximately 3,480,451 minutes (Tasevski). However, this method is a confirmed and completely way of crack a password because every combination of password character is run and tried to match until the password is broken.

3. Hybrid attack works like a dictionary attack, but adds simple numbers or symbols to the password attempt (Tasevski, n.d.). It is a common method of attack utilized by users to change passwords is to add a number or symbol to the end.

C. Salting Password

The reasons that bring many hackers gain successfully in password cracking are firstly, the regularly use of weak password. Secondly, even though the password is stored in the format of hashed value but the successfully used of a pre-computed hashed value of all possible plaintext called Rainbow Table (Boonkrong & Somboonpattanakit, 2016), lead hacker to break the hash of password as well as thirdly, the unused of salt in password storing process (Sriramya & Karthika, 2015).

Salt is a secondary part of word or a set of string that included into password. Salting password is the appropriate solution to protect password. By adding salt into plaintext password then hashed it together, the h(salt, password) make simply secret word becomes longer and more complicated. Therefore, salting password considered as a method to provide a password attack resistant before it is hashed and stored in database server. Value of salt can be applied to use with password in these following ways: prefix salt, suffix salt and inserted salt. Salt value can be generated using the following ways (Kioon, et al., 2013):

1. Using a single salt for example Domain Name System (DNS) with every password kept in the system or database server. Single salt is the simplest way of salting password by selecting one value or set of string which related to the system to use as salt. However, when attacker knows salt value, the entire of password in database can be cracked with no trouble as well. There fore, carefully select the value must be done. The administrator should apply value that is not related to system or easy to guess.

2. Using different random salt value for each password such as time-stamp of individual user when he/she first sign in to the system. This technique provides more protection to password even when 2 or more users use the same password.

The value of salt will be kept separately from password. This solution can also solve the failure of using a single salt however there still be the possibility of attacking password as well.

3. Using an existing value from one column in database as salt. This method is similar to using a single salt thus system administrator must consider the value that will be used as salt thoroughly. As the value of salt exists in database therefore once hackers break into system and know which column is used as salt, the rest of password can be cracked without trouble.

4. Using a variably located in password or database to calculate salt value. As the value used to compute exists in database thus when attackers detect which column and what method used to calculate, they can obtain the attacking as well.

5. Using a variably existed in password or database and information outside to compute salt value. This method provides more safety in salting password process. However, the h (salt, password) still can be broke via brute force attack as well.

Related Work

Kioon, et al., (2013) analyzed the risk of using MD5 hashing in password storing and discussed on several solutions for example adding salt, applying an iterative hashing and introducing a new technique by adding external information which was a random key into salt before encrypting and hashing the password through the MD5 formulation. As a result, the researchers suggested that the used of key stretching create the slowly hashing calculation moreover the used of XOR to generate cipher text make a hashed value infeasible from attack.

Ogini & Ogwara (2014) focused on a database passwords security using a strong hashing algorithm and salting. This work provided a platform to eliminate the need to ever persist user password in plain text or easy to know any users' password. This system was implemented using MD5 algorithm for hashing and a salt pattern. And also introduced the salting pattern for example each password was hashed with a salt that was a randomly numbers generated during the process of calculating the hashed function or using some parts of their log on details. As a result, password with salting pattern provided a stronger security than the original clear text password and also reduced the incident of being cracked and made password stronger when compared with encryption one.

Sriramya & Karthika (2015) discussed on the problem of doing an online shopping that can be vulnerable since the user information and password were kept in plain text in the database and focused on different methods in order to come over the password attack problem by applying the used of strong password, salting, key stretching and iteration hashing as well as chaining method. As a result, the salted password with hashing function using bcrypt algorithm was proposed to provide the security for user when doing an online shopping.

Patel, Patel & Virparia (2013) presented the solution to store password and sensitive data using salt hashing technique. Salt was used to protect password against a dictionary attack also used to reduce the chance of dictionary attack. As a result of this work, salt hash password can prevent the cracking. The attackers have to recalculate their entire dictionary for every individual account when they attempt to crack. However, salting can only against to dictionary attack. The intruder still can attack by any other method for example brute force attack thus salting will not provide must security.

Objectives

1. To enhance the security of weak password.

2. To evaluate if the position of placing salted value has significant to the strength of the password.

Experimental

Hashcat-4.0.1, the password cracking application, is used as password attacking tools. This application is run on computer with Intel (R) Core (TM) i5-7200U CPU @ 2.50GHz 2.71 GHz, Window 64-bit Operating System, x64-based processor and supports NVIDIA GeForce 940MX, 512/2048 MB allocatable, 3MCU. The sample data used in this experimental is the weak passwords collected from many resources that stated top weak password used by user in 2017 (Grauer, 2017; Hern, 2017; Sulleyman, 2017). The reason why these passwords are considered weak is because they are word from dictionary, sequencing number and letter on computer keyboard. All of these words are easy to remember at the same time do not need any memory load therefore it becomes popular for user to select as password when they have to access into many systems. In this experimental, we selected only the password that meet the password policy. The selected secret words must have minimum 8 characters moreover should consists of letter and number. For this reason, there are only 10 passwords used in this study as shown in Table 1.

Table 1 The weak passwords

No.	Password	No.	Password
1	password	6	1qaz2wsx
2	12345678	7	princess
3	123456789	8	qwertyuiop
4	football	9	passw0rd
5	baseball	10	starwars

To make the value of salt and password stronger and harder to attack, the minimum number of salt should be 4 characters (Boonkrong, 2012). The set of characters should involve with both lower and capital letter, numeric and special characters to increase the strong of password as shown in Table 2 (Jense, 2013; Somboonpattanakit & Boonkrong, 2014). Therefore, the value of “z@U1” will be used as single salt with every weak password.

Table 2 List of Password Characters

Type of Data	Characters
Lower Letter	abcdefghijklmnopqrstuvwxyz
Capital Letter	ABCDEFGHIJKLMNOPQRSTUVWXYZ
Numeric	1234567890
Special Characters	!"#$%&'()*+,-./:;<=> ?@[\]^_`{|}~

This salt value is put in front of password as prefix (salt||password) and after password as suffix (password||salt). To insert salt into password, the alphabet frequency as shown in Table 3 will be considered. The alphabet frequency represents the time each alphabet used in word (LetterFrequencies.org, 2016). The alphabet with security rate lower than 0.50 will be considered high risk to crack as it is used regularly thus the salt will be added in front of that character. On the other hand, the alphabet with security rate higher than 0.50 will be considered low risk to attack.

Table 3 Alphabet Frequency

Alphabet	Security Rate	Alphabet	Security Rate
e	0.04	m	0.54
t	0.08	f	0.58
a	0.12	p	0.62
o	0.15	g	0.65
i	0.19	w	0.69
n	0.23	y	0.73
s	0.27	b	0.77
r	0.31	v	0.81
h	0.35	k	0.85
l	0.38	x	0.88
d	0.42	j	0.92
c	0.46	q	0.96
u	0.50	z	1.00

Then original password, salt||password, password||salt and inserted salt were hashed using SHA256 which presently considered as the most security hash function (Aggarwal, Goyal & Aggarwal, 2014) (Grimes, 2015; Raheja, Verma & Raheja, 2014).

Result and Discussion

To consider that salt can improve the security of password, let look at the property of every weak password as shown in Figure 3. Before adding salt with password, there is only "1qaz2wsx" that meet 41% as it is an alphanumeric. While the rest of password are 0% as some of them are lower letter and some are ordering number. After applying salt to password, the property of "123456789" and "qwertyuiop" rapidly increase even it includes sequencing number and letter on computer keyboard. That because the number of character in both of password are 9 characters. The rest of password with salt got 77%. Only "starwars" has 67% that because this word consists of frequency used letter. After inserted salt to the password, there is still word "wars" which exists dictionary left.

Every password was broken with Dictionary Attack for 10 times. The average time used in attacking password and the result were collected from Hashcat as shown in Table 4.

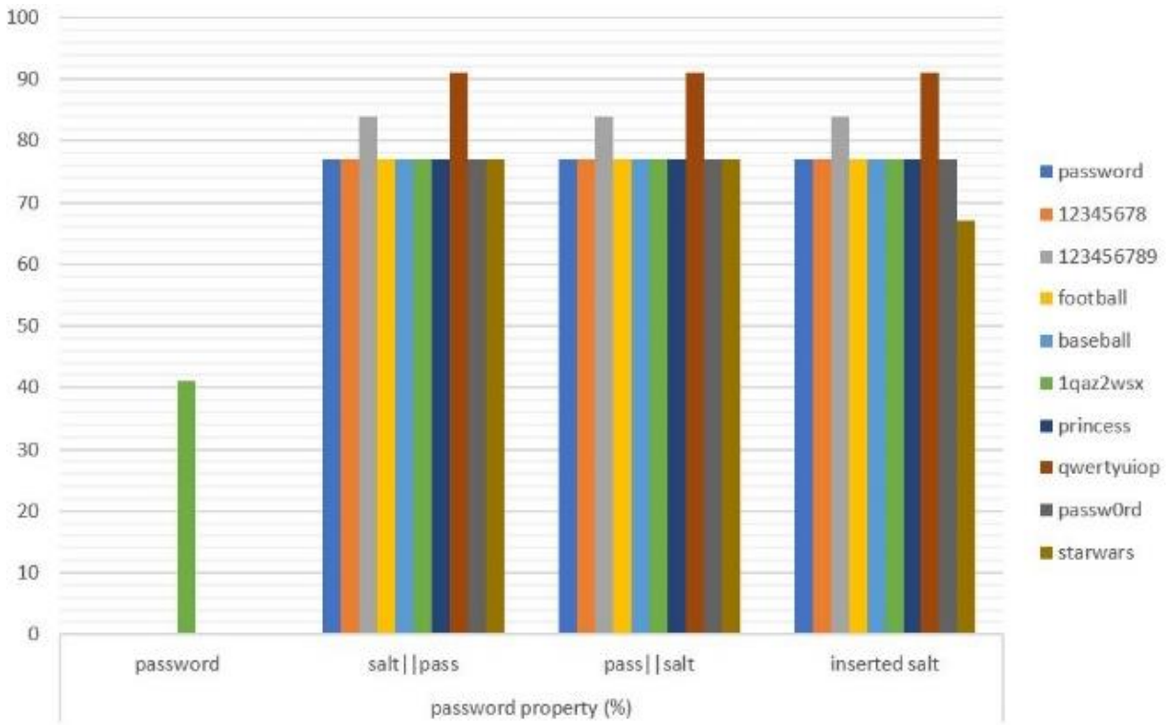

Figure 3 Password Property

Table 4 Time to use in Dictionary Attack/Result

	Time to use in Dictionary Attack (Millisecond)/ Result			
	h(original password)	*h(pass||salt)*	*h(salt||pass)*	*h(inserted salt)*
password	4.68/ Cracked	4.68/ Exhausted	4.57/ Exhausted	4.59/ Exhausted
12345678	4.60/ Cracked	4.58/ Exhausted	4.64/ Exhausted	4.61/ Exhausted
12345678	4.70/ Cracked	4.58/ Exhausted	4.61/ Exhausted	4.58/ Exhausted
football	4.65/ Cracked	4.60/ Exhausted	4.62/ Exhausted	4.58/ Exhausted
baseball	4.62/ Cracked	4.59/ Exhausted	4.61/ Exhausted	4.59/ Exhausted
1qaz2wsx	4.75/ Cracked	4.59/ Exhausted	4.59/ Exhausted	4.57/ Exhausted
princess	4.70/ Cracked	4.61/ Exhausted	4.60/ Exhausted	4.61/ Exhausted
qwertyuio	4.64/ Cracked	4.59/ Exhausted	4.60/ Exhausted	4.60/ Exhausted
passw0rd	4.72/ Cracked	4.58/ Exhausted	4.55/ Exhausted	4.63/ Exhausted
starwars	4.70/ Cracked	4.57/ Exhausted	4.56/ Exhausted	4.63/ Exhausted

From Table 4, the average time of breaking h (original password) with Dictionary Attack is 4.68 Millisecond with the result "Cracked". When consider the "potfile" where all hash passwords that already cracked are stored, there is result of hacking stated in that file. The word in dictionary as well as sequencing number and ordering letter on computer keyboard took shortest time to hack. Even replacing letter with number, it was still cracked within shortly second. From the result, the password which is a two words combination took long time to break via dictionary attack. Finally, the "1qaz2wsx" which is the combination of number and letter provided the most attack resistance however it was still be cracked because it follows the pattern of one digit, one dictionary word which consider as a well-known and simply format used to create password. The average time spending in cracking h (salt||password), h (password||salt) and h (inserted salt) is 4.60 Millisecond. However, the result of attacking is "Exhausted" and when again looked at the "potfile", there is not record available. The word "Exhausted" means that the cracking tool has tried every way to break password but fail. Therefore, by adding salt the hackers must generate a large database of h (password) or wordlist which consumes more time and disk space in order to break the password via dictionary attack. With the limitation of wordlist used in this work, when consider the position of placing salt, it revealed that the time spent in the process of attacking each password become the same range but the password with salt still cannot be cracked.

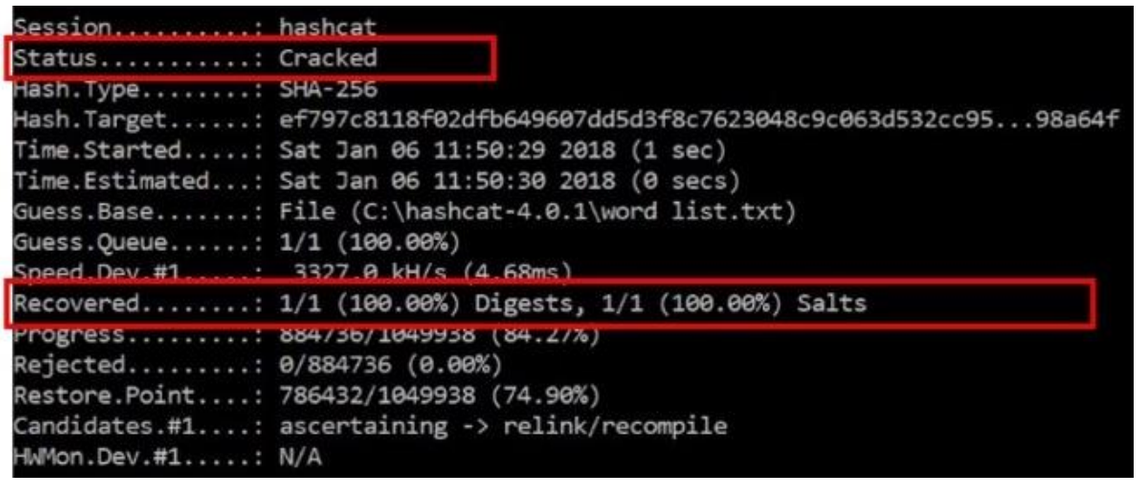

Figure 4 The Result from Cracking Original Password

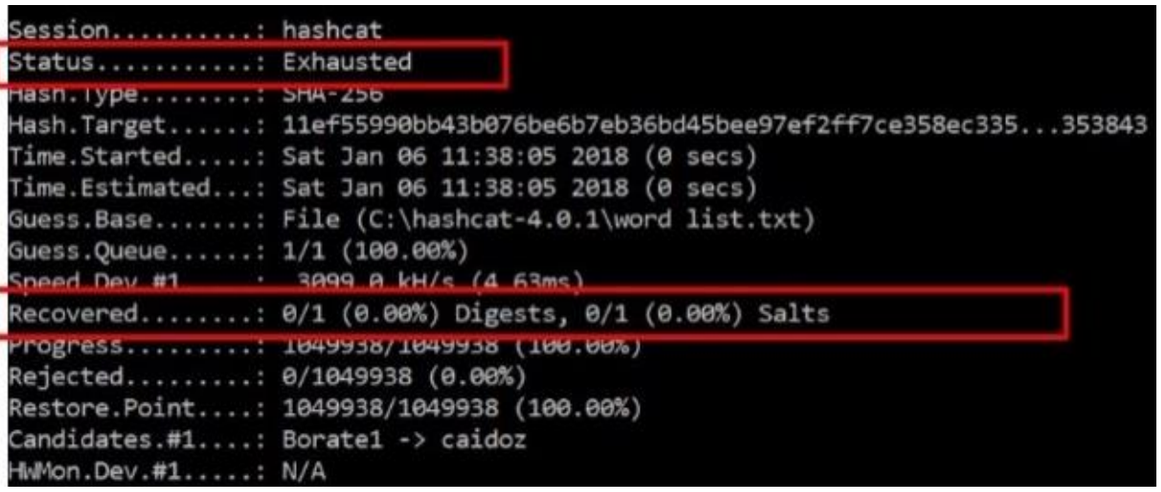

Figure 5 The result from cracking password with salt in three formats

From Figure 4, the h (original password) were cracked without difficulty with recovered "100% Digest" and "100% Salts". This means that the original password was successfully hacked and recovered.

Figure 5 stated the result of hacking h (salt ||password), h (password||salt) and h (inserted salt) is "Exhausted" with recovered "0% Digest" and "0% Salts" which means that the password can not be attacked and recovered due to the limitation of time and wordlist.

Table 5 shows the time using in Brute Force Attack. The password which took longest time to hack is the alphanumeric password and the password that has long characters. By applying salt, the resistance of password attacking for each password is increased for example the password that consists of sequencing number like "12345678" and "123456789".

Table 5 Time to use in Brute Force Attack

	Time to use in Brute Force Attack (Hour)			
	h(original password)	*h(pass||salt)*	*h(salt||pass)*	*h(inserted salt)*
password	0.033333	806,455,440,000	806,455,440,000	806,455,440,000
12345678	0	806,455,440,000	806,455,440,000	806,455,440,000
123456789	0.000278	78,892,380,000,000	78,892,380,000,000	78,892,380,000,000
football	0.033333	806,455,440,000	806,455,440,000	806,455,440,000
baseball	0.033333	806,455,440,000	806,455,440,000	806,455,440,000
1qaz2wsx	0.4	806,455,440,000	806,455,440,000	806,455,440,000
princess	0.033333	806,455,440,000	806,455,440,000	806,455,440,000
qwertyuiop	0.333333	7,109,080,020,000,000	7,109,080,020,000,000	7,109,080,020,000,000
passw0rd	0.4	806,455,440,000	806,455,440,000	806,455,440,000
starwars	0.033333	806,455,440,000	806,455,440,000	245,442,960

The overall time in cracking is too long because minimum number of character used in this test is 8 and each password is a combination of both lower and upper letter, numeric and special characters which cause by salt. The hacker has to perform 958 times to match the password. Thus, adding salt provide more security solution to protect password against Brute Force Attack as it makes weak password longer and more complexed which causes of spend large amount of time in the password attacking process. As follow the condition of inserting salt into password by security rate in Table 3, the password "starwars" which included many frequency letters met the lowest time in password attacking.

The comparison between weak password with salt and strong password in terms of password property, time to crack password via Dictionary and Brute Force Attack is revealed in Table 6.

Table 6 The Comparison between Appied Salt and Strong Password

Password	Password Property	Time to Crack	
		Dictionary Attack	Brute Force Attack
password	0	4.68	806,455,440,000
z@U1password	77	4.68	806,455,440,000
passwordz@U1	77	4.57	806,455,440,000
pazs@sUwo1rd	77	4.59	806,455,440,000
ygda2!E.3u5P	77	4.64	806,455,440,000
i7ovemydog!!	70	4.59	17,531,640,000

The result stated that with the use of salt, weak password becomes strong and hard to attack. In addition, salt can improve the quality of weak password and provide the same level of security in the process of password storing as same as strong password.

Conclusion

This work examined the efficiency of using salt to provide the security of password storing and evaluate if the position of putting salt related to the attack resistance. Adding salt to password before performing hash function enhances the security of weak password that many users who usually select word from dictionary, ordering number and character on computer keyboard to use as their password. Also creating difficulty and complexity to crack password as the attacker has to spend more time or try to apply difference methods to crack the password. The position of placing salt in front of and after the password are not related to the strength of password in this test. However, the inserting salt into password which consist of frequency letter provide lower of the security level than password with no frequency letter. For future work, other type of cracking the password and breaking password with larger set of wordlists should be performed to determine that weak password will still meet the resistance secure as well as the optimization time and resource in protecting password.

References

Aggarwal, S., Goyal, N., & Aggarwal, K. (2014). A review of comparative study of MD5 and SHA security algorithm. *International Journal of Computer Applications, 104*(14), 1-4.

Boonkrong, S. (2012). Security of password. *Information Technology Journal, 8*(2), 112 - 117

Boonkrong, S., & Somboonpattanakit, C. (2016). Dynamic salt generation and placement for secure password storing. *International Journal of Computer Science, 43*(1), 1 - 10.

Chester, J. A. (2015). *Analysis of Password Cracking Methods & Applications.* University of Akron: Ohio's Polytechnic University.

Grauer, Y. (2017). 2016's Worst Passwords Are Just As Bad As 2015's (So Please Tell Me Yours Is Not On The List). Retrieved from https://www.forbes.com.

Grimes, R. A. (2015). All you need to know about the move to SHA-2 encryption. Retrieved from http://www.infoworld.com.

Hern, A. (2017). As easy as 123456: the 25 worst passwords revealed. Retrieved from https://www.theguardian.com.

Jense, B. (2013). 5 Myths of Password Security. Retrieved from https://stormpath.com / blog/5-myths-password-security.

Kioon, M. C. A., Wand, Z., & Das, S. D. (2013). Security Analysis of MD5 Algorithm in Password Storage. *The 2nd International Symposium on Computer, Communication, Control and Automation.*

Letter Frequencies.org. (2016, November). Retrieved from http://letterfrequency.org/

Majumder, J. (2012). Dictionary Attack on MD5 hash. *International Journal of Engineering Research and Applications, 2*(3), 721 - 724.

Ogini, N. O., & Ogwara, N. O. (2014). Securing database passwords using a combination of hashing and salting techniques. *International Journal of Computer Science, 2*(8), 52 - 58.

Patel, P. N., Patel, J. K., & Virparia, P. V. (2013). A cryptography application using salt hash technique. *International Journal of Application or Innovation in Engineering & Management, 2*(6), 236 - 239.

Provos, N., & Mazières, D. (1999). A Future-Adaptable Password Scheme. *The FREENIX Track: 1999 USENIX Annual Technical Conference.*

Raheja, S., Verma, S., & Raheja, S. (2014). Review and Analysis of hashing techniques. *International Journal of Advanced Research in Computer Science and Software Engineering, 4*(5), 292 - 295.

Raza, M., Iqbal, M., Sharif, M., & Haider, W. (n.d.). A survey of password attacks and comparative analysis on methods for secure authentication. *World Applied Sciences Journal, 9*(4), 439 - 444.

SHA. (2016, September). Retrieved from https://www.w3.org

SHA256. (2016, November). Retrieved from http://www.quadibloc.com.

Somboonpattanakit, C., & Boonkrong, S. (2014). Secure Password Storing using Dynamic Salt Selection with Hash Function. *The Tenth National Conference on Computing and Information Technology*, (pp. 240 – 245).

Sriramya, P., & Karthika, R. A. (2015). Providing Password security by salted password hashing using bcrypt algorithm. *ARPN Journal of Engineering and Applied Science, 10*(13), 5551 - 5556.

Stallings, W. (2014). *Cryptography and Network Security Principles and Practices*. (6th ed.) Pearson.

Sulleyman, A. (2017). Most Popular Passwords of 2016 are Desperately Weak Yet Again, Study Finds. Retrieved from http://www.independent.co.uk/news

Tasevski, P. (n.d.). *Password Attacks and Generation Strategies.* n.p.: n.p.