ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645
Indian Journal of Science and Technology, *Vol 10(33), DOI: 10.17485/ijst/2017/v10i33/115252, September 2017*

A Theoretical Framework for Password Security against Offline Guessability Attacks

Shah Zaman Nizamani[1]*, Syed Raheel Hassan[2] and Rafia Naz[1]

[1]Department of Information Technology, Quaid-e-Awam University of Engineering, Science and Technology, Sakrand Road, Nawabshah – 67450, Pakistan; shahzaman@quest.edu.pk, rafia@quest.edu.pk
[2]Department of Computer Systems Engineering, Quaid-e-Awam University of Engineering, Science and Technology, Sakrand Road, Nawabshah – 67450, Pakistan; raheel.hassan@quest.edu.pk

Abstract

Objectives: Security of textual passwords is increased against offline guessability attacks by using different encryption methods. However, even after encryption textual passwords may be guessed through brute-force or dictionary attacks. **Method:** In this paper, a theoretical framework is developed which provides guidelines for improving password security against offline guessability attacks such as brute force and dictionary attacks. In the proposed framework different password security layers are defined which convert a password into a form which is very difficult to crack through offline guessability attacks. The framework layers are implemented at application and database level. **Findings:** In the proposed framework a short and easy to remember password string is converted into a long and random string which does not provide any hint of original password. However, it is important that the methodology or logic used for implementing the framework layers should be hidden from the attackers because the layers' methodology may provide a clue for password cracking. Layers of the proposed framework can be implemented with different logics, which are helpful in hiding the implementation details of the layers. **Application/Improvements:** Proposed framework is not only helpful for improving security of traditional textual password scheme but it can also improve the security for graphical password schemes against offline guessability attacks.

Keywords: Authentication, Guessability Attacks, Privacy, Password Security, Textual Passwords

1. Introduction

Textual password scheme is most widely used technique for user authentication, but this technique has different security and memorability weaknesses. Researchers have proposed many user authentication schemes to overcome the deficiencies of textual password scheme. Some authentication schemes are just the improvement in the user interface of textual password scheme such as S3PAS[1], while other schemes belong to a new category of passwords called graphical password schemes for example cognitive authentication[2]. Both type of schemes generally provides better security than traditional textual password scheme but they are weaker in usability i.e. users require large amount of time for authentication.

Password security attacks may be offline or online, in offline attacks passwords are cracked after hacking the database of an application. While in online attacks the passwords are captured from login screen or through network interception. Brute force and dictionary attacks belong to offline attacks, while shoulder surfing and spyware attacks belong to online attacks.

*Author for correspondence

In order to improve the security against online attacks, all the password elements are need to be indirectly inserted into a login screen. Many authentication schemes allow indirect insertion of passwords, for example in S3PAS scheme[1] alphanumeric characters are randomly shown in the image format. For authentication, a user has to click on the logical triangles formed by the password elements or type characters which belong to each password triangle. Indirect insertion of password helps in improving the security of passwords but it weakens the usability i.e. large effort and time requires for password insertion. Password encryption is used for improving the security of passwords against brute force and dictionary attacks.

In brute force attack a user's password is cracked by comparing all the passwords available inside a password dictionary. A password is cracked when both stored user password string and applied string matches. In brute force attack, large amount of processing and time requires for password cracking.

In dictionary attack a list of passwords is created which has high chances of being set as passwords. The password list may contain previously hacked passwords of different applications or dictionary words. Passwords are cracked through dictionary attack by comparing stored user password string with the dictionary of passwords. In dictionary attack relatively short amount of processing and time is required than brute force attack.

Password cracking through ordinary dictionary attack and brute-force attack is time consuming. An efficient form of dictionary attack is called Lookup Table attacks. In this form of dictionary attack, a list of passwords is saved along with pre-computed hash values. In Lookup Table attack hash values of a password dictionary is compared with a user's passwords. Once hash value of Lookup Table matches with the hashed password string of a user then original password is fetched from the Lookup Table. Lookup Table attacks require less processing and time because runtime generation of hash strings is not required.

Another efficient method of password cracking is called Reverse Lookup Tables attack. In this method, a previously hacked database is used for cracking the passwords of a new database. Before applying this technique hash values and their corresponding password strings of previously hacked database are saved into Reverse Lookup Tables. The computed hash values of Reverse Lookup Tables then compared with the hash values of newly hacked database. Users generally set same password on multiple accounts[3] therefore passwords can be cracked through Reverse Lookup Tables.

Rainbow Table attack is another efficient password cracking method of dictionary attacks. In this method, pre-computed hash values are stored similar to Lookup Tables but passwords and their hash values are saved based upon the length of passwords. Majority of users set passwords of size up to 10 alphanumeric characters[4], therefore in Rainbow Table those password strings are saved whose length may be from 6 to 10 or 12 alphanumeric characters. This intelligent technique of password dictionary generation helps in cracking majority of passwords.

In authentication, cryptographic hash functions[5] such as SHA256 and MD5 are used to protect passwords. Hash functions convert a string into a different set of alphanumeric characters[6]. Hashing may be one-way or two-way, in one-way hashing original data is changed into a new form which cannot be reversed. In two-way hashing original form of data can be generated from hashed string. One of the best techniques for password protection is the use of password hashing through salt scheme[7]. In salt based hashing some alphanumeric characters are added into the original password, so that same password strings could have different encrypted values.

Although one-way hashing greatly improves password security but even though the hashed passwords may be guessed by brute-force or dictionary attacks. Therefore, in the proposed framework some additional measures are suggested along with hashing to improve the password security. In the framework, different layers of password conversion are proposed which works at application and database level. The proposed measures of the framework help in strengthen the security of passwords against offline guessability attacks.

2. Background

User authentication techniques can be divided into three categories based upon the procedure through which credentials are taken from the users[8]. These three categories are discussed here.

a. *Knowledge based authentication:* In this form of authentication a user provides some information (username and password) for identification. A user given information is compared against the stored information or credentials for authentication. Knowledge based authentication technique is most commonly used technique because it is very easy to use and does not require any special hardware for execution. Textual passwords and graphical passwords are the two categories of knowledge based authentication technique.

b. *Token based authentication:* In this form of authentication a user provides a specially designed hardware for identification. In the hardware, authentication information is stored inside a chip. This technique is expensive to execute because a special hardware is required for authentication process. Another concern with this technique is that users have to carry the identification hardware; this can be problematic because the hardware may be theft or lost. In order to further improve the security of token based authentication systems, a password is also need to be provided along with the authentication hardware.
c. *Biometric authentication:* In this technique of user authentication a system identifies through the physical or behavioural characteristics of a user, for example finger print of a user. Although this form of authentication is secure but it is very complex to develop and require special hardware. Privacy is also an issue with biometric based authentication techniques[9].

2.1 Issues in Textual Passwords

In a study[10] researchers found that 87% passwords were consist of only lowercase letters, digits or dictionary words. Based upon the research different password setting restrictions are imposed, in order to create strong passwords.

For better security passwords need to be created with higher entropy and they are also required to be frequently changed.[11] Although both the strategies improve password security but it will affect the memorability of passwords.

An organization named as CSID[4] analysed password habits of American users in 2012. This research shows that password security is the primary concern of users and memorability comes at second number. However, researchers found that 61% people reuse same password in different accounts and mostly users set passwords of size between 8 to 10 characters.

Password creation policies are enforced for strong passwords, for example minimum password length should be eight alphanumeric characters. However, users create weak passwords even after ensuring password creation policies[12].

3. Proposed Framework

A password moves from human mind to application database through different components. In a typical client server architecture password moves through following four components.

1. User Machine/ Client Application
2. Transmission Line
3. Application
4. Database

Security of a user authentication scheme depends upon the security of all the above components. In the proposed framework security measures are suggested at the application and database levels. Password security at first component (user machine / client application) is improved by indirectly inserting the passwords. Password security at second component (transmission line) is improved by using a secure communication protocol such as SSL or TLS[13,14].

Generally passwords are stored into a database through the following steps.

1. Get a password
2. Apply encryption function
3. Save hash value of the password in the database

Password storing through the above mechanism contains security weaknesses against offline attacks. In the proposed framework five layers are identified which works after getting the password of a user up to database storage. A password is stored after making the changes of the five layers. Proposed framework is shown in Figure 1.

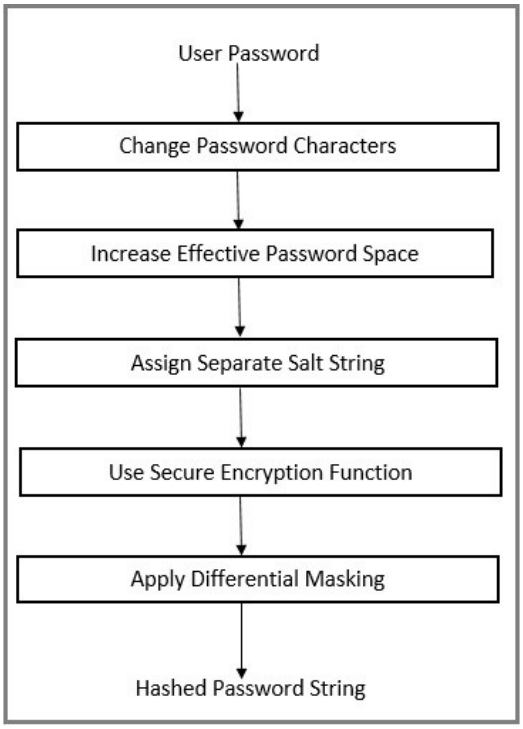

Figure 1. Proposed framework.

3.1 Change Password Characters

Generally, an encryption function is applied to the exact user password and then encrypted value is stored into a database. Passwords of this approach may be cracked by trying all the combinations of strings until matching hash value is reached. In order to overcome this security weakness, alphanumeric characters of a user's password need to be exchanged with different alphanumeric characters as shown in Figure 2.

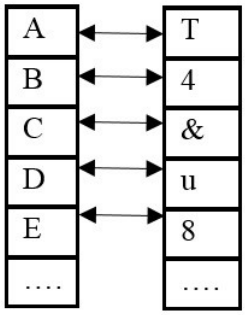

Figure 2. Alphabet mapping.

Figure 2 shows that character "A" will be converted into "T", "B" will be converted into "4" and so on. The information about the conversion of alphanumeric characters should be saved, in order to reuse same conversion method at the time of password matching. Each implementation needs to have different conversations of the alphanumeric characters so that an attacker would not utilize the information of previously hacked application server.

3.2 Increase Effective Password Space

In textual password scheme, theoretical password can be shown through the equation 1 based upon Standard American keyboard.

$$\sum_{i=1}^{94} 94^{i} \quad (1)$$

Theoretical password space contains all the passwords that can be generated from 94 alphanumeric characters in textual password scheme excluding space key. Effective password space is the number of passwords created by the users inside an authentication scheme. Due to memorability issues, users mostly set passwords of size 10 alphanumeric characters or in rare case the password size could reach up to 20 alphanumeric characters. Therefore, effective password space is the number of passwords created from 20 alphanumeric characters. Most applications restrict users to follow password policies such as passwords size should be at-least 8 alphanumeric characters. In such conditions, effective password space of textual password scheme is shown in equation 2.

$$\sum_{i=8}^{94} 94^{i} \quad (2)$$

Due to small size of effective password space, effort for brute-force and dictionary attacks is largely reduced. Therefore, effective password space needs to be increased so that dictionary and brute force attacks become hard to apply.

Effective password space can be increased by converting each alphanumeric character of a password into a list of characters. Table 1 shows the sample conversion of alphabets.

Table 1. Alphabet to string conversion

SNo.	Alphabet	String
1	A	Apple
2	B	Banana
3	C	Cat
…	…	...

Table 1 shows that character "A" is converted into "Apple" and character "B" is converted into "Banana" and so on. In this case the password "AB" becomes "Apple Banana". Hence in case of brute force attack all the strings whose size consist of 11 characters need to be compared rather than strings of size to 2 characters. For security purpose conversion of characters to strings need to be different among different implementations of the proposed layer.

3.3 Assign Separate Salt String

Hashing passwords through salt based encryption technique makes the passwords resilient to Lookup Table and Reverse Lookup Table attacks but salt based hashes can be cracked by dictionary or brute-force attacks. Generally single salt string is used for encrypting all the passwords of an application. Problem with this approach is that once an attacker recognizes the salt string, then other passwords can be easily cracking by using the identified salt string.

Better strategy for salt based encryption is that, each password is encrypted with different salt string. In this approach if an attacker recognizes the salt string of a password, then the salt string cannot be used with other passwords. The attacker has to find out the salt strings of every password.

Separate salt string can be created from many ways such as random or run time salt string generation. In random or static salt string generation, the strings are created once and then they are saved into a database. While run time salt strings can be created by concatenating values of a user's profile such as account creation time, age and gender or any other combination of user profile values. In random salt string generation, the list of salt strings need to be secure. While in run time salt string generation, the process of creating salt strings need to secured.

3.4 Use Secure Encryption Function

Before storing passwords into a database, the passwords need to be encrypted with a secure encryption function. Many encryption functions are available but developers normally do not make enough research for selecting an encryption function. For example, MD5[15] encryption function is still being used, whereas much better bcrypt[16] encryption function is available.

Salt based hashing is a better option for password encryption[17]. Generally, passwords are encrypted with a single salt string. The process of salt based hashing can be further improved by using dynamic salt string i.e. each password is encrypted with different salt string.

3.5 Apply Differential Masking

Generally, passwords are stored in a single database field; due to this approach attackers easily find the hashed password strings after hacking the database of an application. After hacking the database attackers only make efforts in cracking the hash values present in the password field.

In deferential masking hashed password strings are not stored in single field of a table but multiple fields are used for storing hashed password strings. The password strings may be divided into different parts and each part can be stored into a different database field or some useless hash values can be stored in different database fields. Due to differential masking the attackers need to make extra effort for finding actual hashed password string. Differential masking can be implemented with different techniques; one of the techniques is shown in Table 2 for useless password strings.

Table 2. Information stored differential masking

User Id	Password1	Password2	Password3	Password4
1	fs3sdf!@sdff	lkjs*&dsdf	a44asdff%dd	dfsdg@@sd23
2	soidfso322lkj	fs98sdfsdf	we*sdf*23*s*	asdf232bb!
…	….	….	…	…

In Table 2 four password hashes are stored for each user and actual hash string is retrieved by the following steps based upon the flowchart as shown in Figure 3.

1. If remainder becomes 0 after dividing four with user id, then actual hashed password string is present in the column "password4".
2. If remainder becomes 1 after dividing four with user id, then actual hashed password string is present in the column "password3".
3. If remainder becomes 2 after dividing four with user id, then actual hashed password string is present in the column "password2".
4. If above steps fail, then actual hashed password string is present in the column "password1".

A password becomes highly secure after applying the changes of each framework layer and the procedure of each layer is hidden from the attacks.

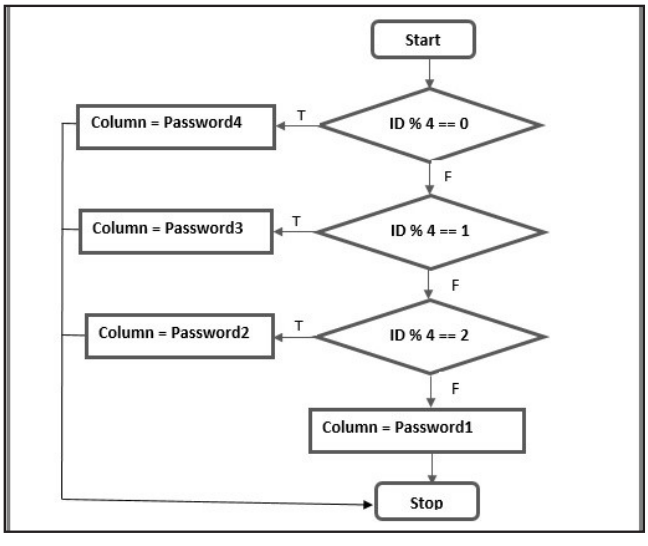

Figure 3. Flowchart for getting password field.

4. Conclusion

Traditional textual password scheme has security weakness against offline attacks and online attacks. Proposed framework is designed for improving the security of textual password and other authentication schemes against offline security attacks. For securing both offline and online attacks, the proposed framework may be used with an authentication scheme which is resilient to online security attacks.

In the proposed framework five layers are suggested for securely storing passwords into a database. Every layer improves the password security but it is not necessary to implement all the framework layers. The proposed framework can be implemented with some selected layers. However, it is important that the layers need to be implemented with different logic on every implementation.

Proposed framework provides security layers at application and database level. Passwords stored with the proposed framework can be cracked when both application and database servers are hacked. For such conditions password security can be maintained when all the framework related processes run on different application server called framework server. The framework server should reply to an IP address where application server is running, in order to hide the implementation logic of the proposed framework.

5. References

1. Zhao H, Li X. S3PAS: a scalable shoulder-surfing resistant textual-graphical password authentication scheme. In the Proceedings of the Institute of Electrical and Electronics Engineers (IEEE) 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW), Canada. 2007 May 21–23; 2:467–72. Crossref.
2. Weinshall D. Cognitive authentication schemes safe against spyware. In Institute of Electrical and Electronics Engineers (IEEE) Symposium on Security and Privacy, USA; 2006 May 21–24. p. 1–6. Crossref.
3. Das A, Bonneau J, Caesar M, Borisov N, Wang X. The tangled web of password reuse. In the Proceedings of the NDSS, San Diego, CA, USA; 2014 Feb 23–26. p. 1–15.
4. Consumer Survey: Password Habits. CSID; 2012. Accessed on 08 January 2017 Available from https://www.csid.com/wp-content/uploads/2012/09/CS_PasswordSurvey_FullReport_FINAL.pdf
5. Preneel B. Cryptographic hash functions. Transactions on Emerging Telecommunications Technologies. 1994 Jul; 5(4):431–48. Crossref
6. Coron JS, Dodis Y, Malinaud C, Puniya P. Merkle-Damgard revisited: how to construct a hash function. In the Proceeding of the Annual International Cryptology Conference, Lecture Notes in Computer Science, Springer. 2005; 3621:430–48. Crossref
7. Klein DV. Foiling the cracker: a survey of and improvements to, password security. In the Proceedings of the 2nd USENIX Security Workshop; 1990. p. 5–14.
8. Suo X, Zhu Y, Owen GS. Graphical passwords: a survey. In the Proceedings of the Institute of Electrical and Electronics Engineers (IEEE) 21st Annual Computer Security Applications Conference, USA; 2005 Dec 5–9. p. 1–10.
9. Schneier B. Inside risks: the uses and abuses of biometrics. Communications of the Association for Computing Machinery (ACM). 1999 Aug; 42(8):136. Crossref
10. Morris R, Thompson K. Password security: a case history. Communications of the Association for Computing Machinery (ACM). 1979 Nov; 22(11):594–7. Crossref
11. Cisar P, Cisar SM. Password-a form of authentication. In the Proceedings of the Institute of Electrical and Electronics Engineers (IEEE) 5th International Symposium on Intelligent Systems and Informatics, Serbia; 2007 Aug 24–25. p. 29–32. Crossref
12. Zviran M, Haga WJ. Password security: an empirical study. Journal of Management Information Systems. 1999; 15(4):161–85. Crossref
13. Dierks T. The transport layer security (TLS) protocol version 1.2. Rescorla E editor, RTFM Inc; 2008 Aug. p. 4–63.
14. Freier A, Karlton P, Kocher P. The secure sockets layer (SSL) protocol version 3.0. Internet Engineering Task Force (IETF); 2011 Aug. p. 5–67.
15. Xiao-ling W. Research and application of MD5 encryption algorithm [J]. Information Technology; 2010.
16. Provos N, Mazieres D. Bcrypt algorithm. USENIX; 1999 Apr 28. p. 1–13.
17. Florencio D, Herley C, Oorschot PCV. An administrator's guide to internet password research. In the Proceedings of the Association for Computing Machinery (ACM) 28th USENIX conference on Large Installation System Administration (LISA), WA; 2014 Nov 9–14. p. 35–52.