Password Security Scheme based on a Honeywords Generation through Hashing Key and Salting Process

Teena Kashyap[1] Er. Divya[2]
[1]Research Scholar [2]Assistant Professor
[1,2]RPIIT Bastara, Karnal

Abstract— The concept of authentication has evolved in the recent past from simple password based authentication to face recognition, voice recognition and biometric based authentication and so on. But, even after all these various evolutions, password based authentication remains the most widely used method. So, password protection schemes become very important in such cases. This has been a man area of research ever since the inception of word "Authentication". The researchers, in recent past, have deployed different techniques based on salting, hashing, quantum cryptography and split key methods. The base research by Binoj Koshy [4] describes a salting based hashing process called Chameleon Salting Hashing where he successfully implemented a password security scheme to prevent various attacks. In this paper, we propose an advanced hashing technique combined with salting to improve the security of passwords even though the intruder gets an access to the database. The hashing technique is based on some simple bit conversion methods making it highly efficient in terms of time consumption. The method involves creation of a blacklist as a result of many hashing processes conducted on the input password. Blacklist also known as honeywords list. Each hashing process is based on a different hashing key. The hashing key defines the sequence of actions defined in the hashing process, that take place on the input password. The base research had a major drawback in that it could not prevent brute force attack and also the salting based hashed technique was complicated and the time complexity was high for this. The proposed methodology has far lesser time complexity because it is based on simple bit operations in a 5 step process. The final results shown both numerically and graphically clearly demonstrate the efficiency of proposed technique against base methodology.

Key words: Blacklist, Salting, Hashing, Authentication

I. INTRODUCTION

Authentication is one of the most important requirements to secure information. There exist various methods for authentication (e.g. passwords, PINs etc). Among these existing methods password based systems are easier to implement and most generally used method for authentication. But it is a blunder mistake of application developers to store user passwords within databases as plaintext or only with their unsalted hash values. Many successful hacking attempts that enabled attackers to get unauthorized access to sensitive database entries including user passwords have been practiced in the past[1]. In the past, many different techniques based on salting, hashing, quantum cryptography and split key methods have been proposed.

In this thesis, I work towards creating a technique where it almost becomes impossible for the intruder to break the security and also. If intruder has unauthorized access to the data file, then the system will generate an alarm to identify the attacker. We provide here a technique which creates a list of passwords called as blacklist using different permutations of current password and the correct password in the list is not known to any user.

II. LITERATURE SURVEY

Remarkable work has been already done in the field of password security schemes. Here we review some of the previous studies and researches related to our work of password security.

Warutet. al in [1] carried on his research to mitigate the issue of weak passwords by proposing a context-based password strength meter. The authors conducted a randomized experiment on Amazon MTurk and observed the change in users' behavior. The results showed that the proposed method was significantly effective. Users exposed to this password strength meter were more likely to change their passwords after seeing the warning message, and those new passwords were stronger. Furthermore, users were willing to invest their time to learn about creating a stronger password, even in a traditional password strength meter setting. The findings suggested that simply incorporating contextual information to password strength meters were an effective method in promoting more secure behaviors among end users. The study showed that providing additional contextual information along with warning messages displayed by password strength meters could enhance understanding among users, resulting in improved password generating behaviors.

Katha Chanda in [5] dealt with password security, a close look at what went into making a password strong and the difficulty involved in breaking a password. The first few sections discussed related work and proved graphically and mathematically the different aspects of password securities, overlooked vulnerabilities and the importance of passwords that were widely ignored. This work described tests that were carried out to evaluate the resistance of passwords of varying strength against brute force attacks. It also discussed overlooked parameters such as entropy and how it tied in to password strength. This work also discussed the password composition enforcement of different popular websites and then presented a system designed to provide an adaptive and effective measure of password strength. This paper contributed toward minimizing the risk posed by those seeking to expose sensitive digital data. It provided solutions for making password breaking more difficult as well as convinced users to choose and set hard-to-break passwords.

III. PROBLEM FORMULATED

A. Base Research

Chameleon Salting[4] is an innovative initiative BY Binoj Koshy et. al to enhance security to the end user, where the end user is provided a secure environment for entity authentication even without the user having to implement or change the way the application is being used. The Salted password is a password generated after the user completes the process of entity authentication; in which the user input 'Password' is appended with a fixed length hash value called the 'Salt'. Chameleon Hash Function is a 'Collision-Resistant' Hash Function. The algorithm produces Non-transferable Signatures called 'Chameleon Signatures'. It is a cryptographic hash function, which includes characteristics like, Non-repetitive pre-image, Collision-Resistant and Pseudo-randomized generation. The chameleon hash conceals the 'Public Key' by attaching the Function with a corresponding unique trapdoor. The solution proposed can be implemented with least effort and with no additional cost; hence the project implementer will be able to convince the management hierarchy to adopt the 'Chameleon Salting' solution. the exploitation of brute-force attack is still possible on salted passwords. Notwithstanding the fact; that there exists a limited protection threshold against dictionary attacks. 'Chameleon Salting' is only activated when the user 'Logs-in' into the application.

Querying data in relational databases is often challenging. SQL is the standard query language for relational databases. While expressive and powerful, SQL is too difficult for users without technical training. Even for users with expertise in programming languages, it can be challenging because it requires that users know the exact schema of the database, the roles of various entities in a query, and the precise join paths to be followed. As the database user base is shifting towards non-experts, designing user-friendly query interfaces will be a more important goal in database community. So, to fulfill the needs of these non-technical people, we need a way to convert their natural queries into SQL queries and fetch desired results.

B. Scope of Proposed System

The scope of the proposed system is as follows:

- To work with RDBMS one should know the syntax of the commands of SQL.
- The interface language is chosen to be English for accommodating wider users.
- Input from the user is taken in the form of questions (like what, who, where).
- All the values in the input natural language statement have to be in double quotes which yield to identify the values from the user input statement.
- A limited data dictionary is used where all possible words related to a particular system will be included. The data dictionary of the system must be regularly updated with words that are specific to the particular system.
- Split the question string in to tokens and give order number to each token identified.
- To remove excessive words from the user input statement. Escape words have been considered which must be regularly updated with words that are specific to the particular system.
- To construct an RDBMSQL query using tokens, an algorithm has to be developed.
- Ambiguity among the words will be taken care of while processing the natural language.

IV. PROPOSED METHODOLOGY

The proposed methodology is a hybrid concept of salting and a simple but secure hashing technique to finally generate a password protection scheme which is not vulnerable to any of the attacks including brute force attack which the base research could not counter. The process can be divided into following steps.

1) Step 1: User enters the password into the password field.
2) Step 2: The salting is done on this password to add some bits to the right and left of the entered password.
3) Step 3: The salted password is then taken through the 5-step hashing process using multiple hash keys which are generated randomly.
4) Step 4: The final result generated is the hashed string which is matched with the hashed string blacklist retrieved from the database.
5) Step 5: If the generated hashed string matches the middle element in the blacklist then, the user is authenticated.
6) Step 6: If the user enters the password which matches one of the elements in the blacklist, then it denotes an attack and the system generates the alarm.
7) Step 7: Any unsuccessful match with any element in blacklist causes an unsuccessful login.

The steps involved in the signup process are as follows:

1) Step 1: The user preferred password is taken through the salting and hashing process detailed below.
2) Step 2: Different hash keys are used to generate a list of hashed strings which collectively joined to form what is called blacklist. This blacklist contains the middle element as the user password hash string.
3) Step 3: This blacklist is stored into the database for the particular user.

A. Salting Process

SALT(str)
Step 1. Let L = length of string str
Step 2. N = L/8
Step 3. Generate a binary string formed by taking nth bits from left side.
Step 4. Now put the first four bits at the left end of str and last four bits to the right end of str.

B. Hashing Process (str)

The hashing process is sequence of 5 basic bit level operations. The process is sequenced as under:

1) Step 1. A Step: This step converts input string into binary string

2) Step 2. N1 Step: The value of N1 is used in this step. The N1 character from either side is interchanged.
3) Step 3. R Step: This step involves rotation in left direction for the input string R times.
4) Step 4. N2 Step: This step involves XORing of input binary string with a binary string of equal length formed by taking a sequence of N2 0s and N2 1s.
5) Step 5. A- Step: This is reverse of the initial step which means it converts binary string into the text string.

These steps can be done with a changed sequence to generate a different resultant hashed string every time. This forms the basis of our main concept called honeyword (Blacklist) generation.

V. RESULTS

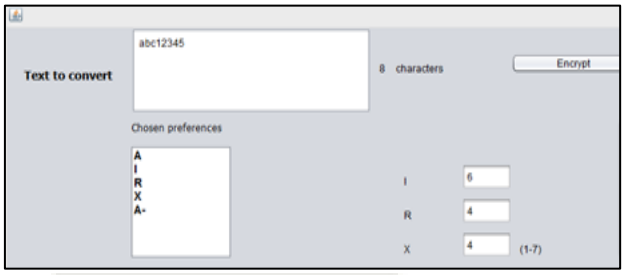

Fig. 4: convert input string into hashed string

The figure above demonstrates how the exact hashing process goes. The text to be hashed is written at the top box. The operations 12345 are sequenced as AIRXA-. The values of I and R stand for the values for N1 and N2 in the hashing algorithm. The values of X denote the value of R1. The minimum length required for password is 6 chars.

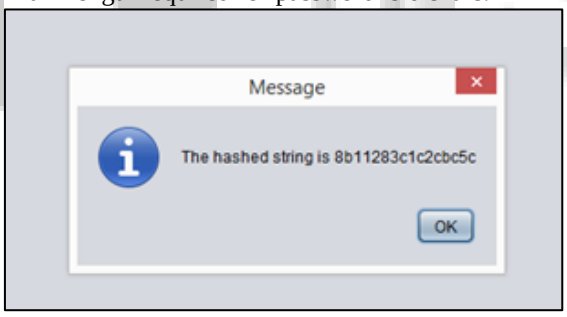

Fig. 4.2: The hashed string generated as a result of salting and then hashing

The fig.4.2 is in continuity with the fig.4.1. In the previous figure when we click on encrypt to convert that into hashed string, then the hash string is shown as above.

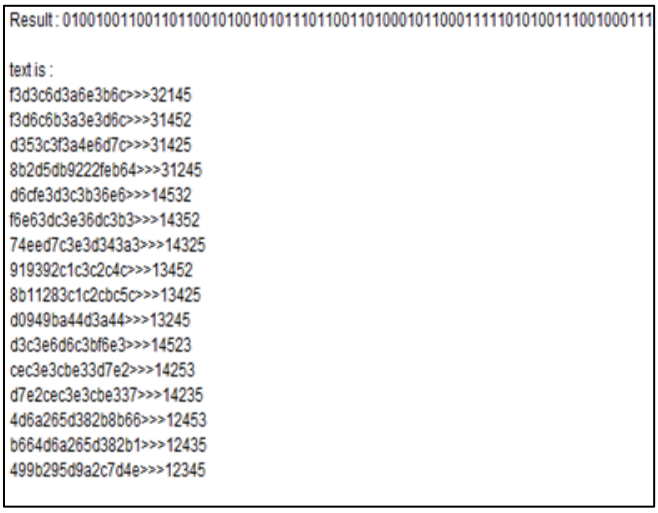

Fig. 4.3: Different hashed strings resulted from conversion according to different hashing keys to generate blacklist

As we have seen and understood the process of hashing, so the figure above shows the result of hashing on the entered password. The sequence of different hexadecimal hashed strings generated is a result of the different hashing operations applied in different sequences taken from the hash keys generated by the system.

f3d3c6d3a6e3b6c, f3d6c6b3a3e3d6c, d353c3f3a4e6d7c, 8b2d5db9222feb64, d6cfe3d3c3b36e6, f6e63dc3e36dc3b3, 74eed7c3e3d343a3, 919392c1c3c2c4c, 8b11283c1c2cbc5c, d0949ba44d3a44, d3c3e6d6c3bf6e3, cec3e3cbe33d7e2, d7e2cec3e3cbe337,4d6a265d382b8b66, b664d6a265d382b1, 499b295d9a2c7d4e

f3d3c6d3a6e3b6c>>>32145
f3d6c6b3a3e3d6c>>>31452
d353c3f3a4e6d7c>>>31425
8b2d5db9222feb64>>>31245
d6cfe3d3c3b36e6>>>14532
f6e63dc3e36dc3b3>>>14352
74eed7c3e3d343a3>>>14325
919392c1c3c2c4c>>>13452
8b11283c1c2cbc5c>>>13425
d0949ba44d3a44>>>13245
d3c3e6d6c3bf6e3>>>14523
cec3e3cbe33d7e2>>>14253
d7e2cec3e3cbe337>>>14235
4d6a265d382b8b66>>>12453
b664d6a265d382b1>>>12435
499b295d9a2c7d4e>>>12345

Fig. 4.4: Generated password to be stored into database

In the figure 4.4 as shown above, we see the different hashed strings generated as a result of hashing applied on password entered by user. The hashing operations are implied by the sequence like"32145". All the hashed strings are joined together to form a list of hashed hexadecimal strings called BLACKLIST. The BLACKLIST is shown at the top of the figure. This is stored in the database to misguide the intruder if he gets access to the database somehow.

Taking sequence 32145
Rotating text -4 steps
After R : 2345abc1

Interchanging position 6(b) and 3(4)
After I step: 23b5a4c1
'0011001000110011011000100011010101100001001101000110001100110001' to binary:
0000000000110010001100110110001000110101011000010011010001100110001
XORing.....
String 1:
0000000000110010001100110110001000110101011000010011010001100110001
String 2:
0000111100001111000011110000111100001111000011110000111100001111
Result:
0000111100111101001111000110110100111010011011100011101101100000111110
text is:
Taking sequence 31452
Rotating text -4 steps
After R: 2345abc1
'0011001000110011001101000011010101100001011000100110001100110001' to binary:
0000000000110010001100110011010000110101011000010011000101100110001
XORing.....
String 1:
0000000000110010001100110011010000110101011000010011000101100110001
String 2:
0000111100001111000011110000111100001111000011110000111100001111
Result:
0000111100111101001111000011101100111010011011100011000101100110001
text is:
Interchanging position 6(3) and 10(6)
Interchanging position 12(6) and 4(3)
After I step: f3d6c6b3a3e3d6c

Fig. 4.5: Different steps involved in hashing and salting based on different Hash keys

The figure 4.5 shown above displays the sequence of steps happening on the input password. The result generated after each step is shown in the figure. The sequence is specified by the hash key.

Fig. 4.6: Alarm generated by system for an intruder who has access to database

This is the main aspect of the proposed system shown above figure. The main point in this research is not just to perform safe authentication but also be able to detect when an intruder tries to login. The intruder can try to hack the system in two ways - by random passwords generation and login attempts. This can easily be encountered by putting a check on the maximum login attempts. The other way can be if intruder is able to access the database. In that case if he uses the blacklist or any part of blacklist as a password then the system generates an alarm for intrusion happening.

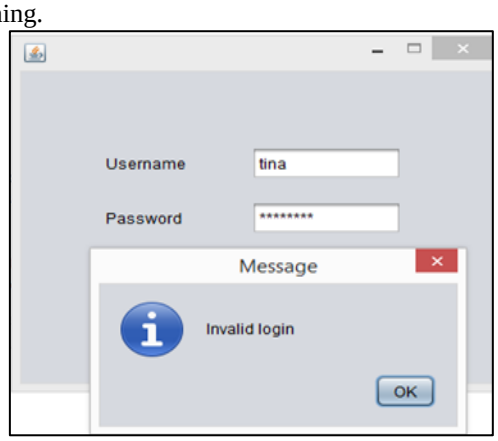

Fig. 4.7: Invalid login attempted

The figure 4.7 shown above displays the response generate by system when you enter an invalid password. This is normal authentication error.

Length Of Passwords (in characters)	Time Complexity (in milliseconds)
6	46 ms
8	31 ms
10	47 ms
12	32 ms
Grand Total	156 ms

Table 1: Time taken for different password lengths

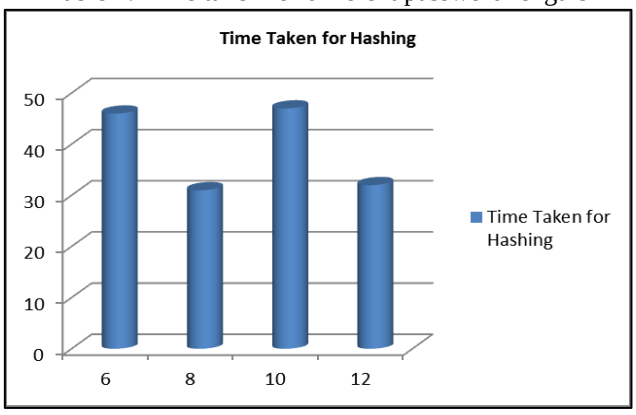

Chart 1: Length of password vs time taken for hashing

The tabular data and the graph above show that the time taken for hashing is not dependent on the length of password for the normal lengths like 6-12 chars. We didn't check this for greater lengths like 100 or 200 chars because those sound like some improbable lengths for passwords. The independence of time is mainly because the bit conversion hashing employed here involves basic bit operations so the time involved is so small that the process scheduling time lapses take over the minute time involved in conversion.

VI. CONCLUSION

In this Research, we propose an advanced hashing technique combined with salting to improve the security of passwords even though the intruder gets an access to the database. This is Hybrid concept of hashing and salting process. We also combine Chameleon salting with Honeywords list generation (Blacklist in this case). The hashing technique is based on some simple bit conversion methods making it highly efficient in terms of time consumption. Hashing technique is the 5 step hashing process using multiple hash keys. The different result (Hashed String) generated from the blacklist in the password column. This creates a camouflage for the users. In the way, he thinks the blacklist to be the password or one of the members of blacklist as password. In both cases, system generates alarm indicating the presence of intruder (Hackers into database). The final results shown both numerically and graphically clearly demonstrate the efficiency of proposed technique against base methodology. This scheme is safe against brute force attack and password guessing method unlike its predecessors.

REFERENCES

[1] WarutKhern-am-nuai, Weining Yang and Ninghui Li, "Using Context-Based Password Strength Meter to Nudge Users' Password Generating Behavior: A Randomized Experiment", 50th Hawaii International Conference on System Sciences | ieee 2017
[2] Steffen Werner, Christopher Hauck, and Marshall Masingale, "Password Entry Times for Recognition-based Graphical Passwords", Human Factors and Ergonomics Society 2016 Annual Meeting, ieee 2016
[3] Bruno Korbar, Jim Blythe, Ross Koppel, Vijay Kothari and Sean Smith, "Validating an Agent-Based Model of Human Password Behavior", AAAI Conference on Artificial Intelligence, Artificial Intelligence for Cyber Security: IEEE2016.
[4] BinojKoshy, NilayMistry and Khyati Jain, "Chameleon Salting: The New Concept ofAuthentication Management", IFS annual Journal 2016.
[5] Katha Chanda, "Password Security: An Analysis of Password Strengths and Vulnerabilities", I. J. Computer Network and Information Security, 2016, 7, 23-30
[6] Blasé Ur, "Supporting Password-Security Decisions with Data", PNC Center for Financial Services Innovation and Microsoft Research. 2016.
[7] Blake Ross, Colin Jackson, Nick Miyake, Dan Boneh and John C. Mitchell, "Stronger password authentication using browser extensions", NSF Science Project 2005.
[8] R.V.Sudhakar, A. Mruthyunjayam, D. Suguna Kuamari, M. Ravi Kumar and B.V.S. Ramesh Babu, "Improving Login Authorization by Providing Graphical Password (Security)", Int. Journal of Engineering Research and Application, Vol. 3, Issue 6, Nov-Dec 2013, pp.484-489
[9] Joseph Bonneau, Cormac Herley, Paul C. Van Oorschot and Frank Stajano, "Passwords and the Evolution of Imperfect Authentication", Communications of the ACM vol. 58 no. 7, July 2015 pp. 78–87.
[10] Muhammad Adeka, Simon Shepherd and RaedAbd-Alhameed, "Resolving the Password Security Purgatory in the Contexts of Technology, Security and Human Factors", International Conference on Computer Applications Technology (ICCAT),2015.
[11] Blase Ur, Sean M. Segreti, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, SarangaKomanduri, Darya Kurilova, Michelle L. Mazurek, William Melicher and Richard Shay, "Measuring Real-World Accuracies and Biases in Modeling Password Guessability", 24th USENIX Security Symposium August 12–14, 2015
[12] Imran Erguler, "Achieving Flatness: Selecting the Honeywords from Existing User Passwords", IEEE Transactions on Dependable and Secure Computing 2015
[13] Harikrishan T and C Babu, "Cryptanalysis of Hummingbird Algorithm with Improved Security and Throughput", International Conference on VLSI Systems, Architecture, Technology and Applications, IEEE 2015
[14] YevgeniyDodis, SiyaoGuo and Jonathan Katz, "Fixing Cracks in the Concrete: Random Oracles with Auxiliary Input, Revisited", NSF-REU 2015.
[15] Emin ˙Islam Tatlı, "Cracking more Password Hashes with Patterns", IEEE Transactions on Information Forensics and Security 2015.
[16] PoojaKolte, ReshmaGutal and PriyankaBhairat, "An Efficient Password Security Mechanism Using Two ServerAuthentication and Key Exchange", IJARCSMSVolume 3, Issue 1, January 2015
[17] SeyedHasan, MortazaviZarch, Hussein Soltani and madihesadatYazdani, "Enhance The Security Of Password By Fuzzy Controller", IEEE 2014
[18] Florence Mwagwabi, Tanya mcgill and Michael Dixon, "Improving Compliance with Password Guidelines: How User Perceptions of Passwords and Security Threats Affect Compliance with Guidelines", 47th Hawaii International Conference on System Science, IEEE 2014.
[19] Anupam Das, Joseph Bonneau, Matthew Caesar, Nikita Borisov and xiaofengWang,"The Tangled Web of Password Reuse", NDSS 2014,23-26 February 2014,San Diego,CA,USA, Internet Society,ISBN 1-891562-35-5.
[20] Yulong yang, jannelindqvist and anti oulasvirta, "Text Entry Method Affects Password Security", LASER 2014, USENIX Association.
[21] Samuel OjodeOluoch, "Improving password security using location based intelligence", International Journal of Scientific and Research Publications, Volume 4, Issue 2, February 2014
[22] SeyedHasanMortazaviZarch, Hussein Soltani and madihesadatYazdani, "ENHANCE THE SECURITY OF PASSWORD BY FUZZY CONTROLLER", 978-1-4799-3351-8/14 2014 IEEE.
[23] Ding Wang and Ping Wang, " Offline Dictionary Attack on Password Authentication Schemes using Smart Cards", 16th Information Security Conference (ISC 2013), November 13-15, 2013, Springer--Verlag, pp.1-16.

