Achieving Security using Honeyword

Shubham Kute
B.E Student, Dept. of Computer Engineering, D.Y.Patil Institute of Engineering & Technology, Ambi, Pune University, Maharashtra

Vrushali Thite
B.E Student, Dept. of Computer Engineering, D.Y.Patil Institute of Engineering & Technology, Ambi, Pune University, Maharashtra

Sharmila Chopade
Professor, Dept. of Computer Engineering, D.Y.Patil Institute of Engineering & Technology, Ambi, Pune University, Maharashtra

ABSTRACT

An expose of the password file is a serious security problem. The research shows that system uses encrypted form to store the original password. Jewels and Rivest proposed “Honeyword” to detect attacks against the hashed password database. Authorized password is stored with several honeywords for every user. The attacker who has stolen hash password file cannot be sure whether it is the real password or a Honeyword for an account, even if honeyword is selected properly. Entering a Honeyword to login will notify the administrator by sending the message about the breach of the password file. As the admin receives the message of the breach, the IP gets blocked for a particular time and also tries to find the location of the IP address. For the generation and encryption of the Honeyword, two encryption techniques are used such as the Salt method for encryption and “sha256” algorithm. Although the approach selects the honeywords from existing user passwords in the system.

Keywords

Authentication, Cryptography, Honeypot, Password-cracking.

1. INTRODUCTION

Security is an important factor which hides the personal data from intruders. An exposal of the password file is a security problem that may affect millions of the users and companies. Passwords are a notoriously weak authentication mechanism. Poor passwords are chosen frequently by the users. “Honeyword” is one of the methods to identify the occurrence of password disclosure. In honeyword approach, administrator dynamically create a fake user account to detect password disclosure, if any one of the honeypot password file is exposed. For example, the LinkedIn passwords were using the SHA-1 algorithm without a salt and similarly the passwords in the eHarmony system were also stored using unsalted MD5 hashes [6,11].

Honeyword system includes an auxiliary secure server, which is called "honeychecker" which can find a real password of user among number of honeywords and immediately notify the user whenever honeyword is used by attacker. The honeyword system suggest improvements for generation of number of honeywords per user and managing real passwords from analyzing previous system. In order to increase the number of unique password in system, i.e reduce common passwords, users should be forced to adhere to a password-composition policy like basic8, basic16 [4] Furthermore, we point out that the key item for salt method is the generation algorithm of the honeywords such that they shall be indistinguishable from the correct passwords. Therefore, we propose a new approach that uses passwords of other users in the system for honeyword sets, i.e., realistic honeywords are provided. Moreover, salt technique also reduces the storage cost compared with the honeyword method [11]. The generation of the password uses “salt” encryption with “sha256” technique. Hex is also one of the technique which is used for creation of honeyword. The attacker cannot login to system without a high risk of being detected even if the attacker has broken all the hashes in the password file. For easy understand the difference between cracking a single password and a set of them, consider a single password file that contains hundreds of usernames and hashed passwords. An attacker could compute hash and then check whether that hash appears anywhere in the file without salt. The likelihood of a match, i.e. cracking one of the passwords with that attempt, increases with the number of passwords in the file. If salts are present, then the attacker would have to compute hash, compare against entry A then hash compare against entry B, and so on.

2. LITERATURE SURVEY

1. “Honeywords: Making Password-Cracking Detectable” [5], A. Juels, R.L. Rivest, In this paper, the author proposed a simple method for improving the security of hashed password i.e “honeywords” (false password) cooperate with every user’s account, so the attacker will not get whether he has found the password or honeyword. The attempted use of a honeyword for login sets off an alarm. The system distinguishes real usernames from fake usernames and thus avoid detection. Someone who steals a password file can brute force to search for password, even if honeyword are used.

2. “Improving security using deception”[8], M.H.Almeshekah, E.H.Spafford and M.J.Atallah, In this paper, the author mentioned how information can be protected and show how system can structure methods to achieve better results. System analyzes complex relationships among protection techniques. The goals are to prevent unauthorized access to information stored in our system and hide existing nature of system. The deception technique such as, Honeypot which is an information system resource whose value lies in unauthorized use of that resources. Disadvantage of honeypot is that honeypots are largely useless against insiders who are aware of existence of the honeypots. To protect digital Information some methods are used like denial isolation using novel taxonomy.

3. “Examination of a new defense mechanism honeywords”[9], Z.A.Genc, S.Kardas and M.S.Kiraz, In this paper, the author analyzes the honeyword system according to both functionality and the security. The author suggests improvement for number of honeywords per user, generating typo-safe honeywords and managing old passwords and thus get

solution to the open problem of active attacks. Author also gives the brief information about the numerous attacks to obtain a user's password. In this paper author present four distinct solutions for improvement of honeyword system, from that three solutions are proposed to make the system more robust and the last solution will deal with an active attack. The honeyword system is powerful defense mechanism. Suppose the attacker has broken all the hashes in the password file, still cannot login in the system without a high risk of being detected.

4. "Password cracking using Probabilistic Context-free Grammars"[3], M.Weir, S.Aggarwal, B.Medeiros, Bill Glodek, This paper describes a new method that generates password structures in highest probability order. The system will automatically create a probabilistic context-free grammar based upon a training set of previously disclosed password. In off-line password recovery, the attacker typically possesses only a hash of the original password. The attacker makes a guess as to the value of the original password to crack it. The attacker then hashes that guess using the appropriate password hashing algorithm and compares the two hashes. If the two hashes match, the attacker has discovered original password.

5. "Achiving Flatness by Selecting the Honeywords from existing user passwords"[11], S.P.Khedkar, B.Bachhav, P.Parsewar, R.Tirmal, In this paper the author mentioned about the use of authorization as the username and password checking is much more important in the security system. To protect real password from third party, real password is converted to new password using honeyword. Providing number, text, special characters, validation passwords are most likely used authentication methods. The password composition policies make password difficult to think. Generation algorithm of the honeywords are used and new method is used that create the honeywords using the existing user passwords combination in hash format. Security of honeyword system is more important and also introduce a number of defects that need to be fixed before successful realization of the scheme. Honeyword's system directly depends on the generation algorithm.

6. "A Large-Scale Study Of Web Password Habits"[1], Dinei Florencio and Cormac Herley, In this paper authors report the results of large scale study of password use and password re-use habits. How many accounts each user has can be estimated, how many times the user is entering the password per day, how often the password in shared among different networking sites and how often they are forgotten. All social networking sites and email, banks etc., require passwords. It is so important that HTML has a special form field to allow for the special treatment they require. As there are certain limitations of the data, user may type passwords from more than one machine and thus miss potentially large fraction of their password behavior. Authors can be able to estimate the number of passwords they type per day and the phishing percent of hacker in the overall population.

3. SYSTEM IMPLEMENTATION

A substitute approach that selects the honeywords from existing user passwords in the system in order to provide realistic honeywords, a perfectly flat honeyword generation method.

The Honeyword system has following modules:

A. Architecture Diagram:

a) User Registration (Sign In / Sign Up) : The module contain the user first name, last name, email_id, password and mobile_number. For the first time registration user need to fill this details.

b) Bank Applications : The users personal information is stored.

c) Honeywords Generation : Generating the fake password using salt and sha256 encryption method.

d) Honeyword Checker : The checker compare the entered password with original password to detect the unauthorized user.

e) Block IP Address : After three unsuccessful attempts the IP get blocked.

3.1 Technical Descriptation context:

If we assume a computer system with n users U1, U2,..., Un; here U_i is the username for the i^{th} user. We let P_i denote the password for user U_i. This is the correct password; it is what user U_i uses to log in to the system. The system uses a cryptographic hash function which is H and stores hashes of passwords rather than raw password.

Attack scenarios:

There are many attack scenarios relating to passwords hacking.

Denial-of-Service Attack(DoS):

In the DoS attack single system is targeted by multiple system to compact its performance. Dos attacks manage this by flooding the traffic or sending information to target that triggers a crash. System uses CAPTCHA or a similar mechanism to prevent automated login attempts and adversary is patient to try all guesses manually.

Brute-Force Attack:

It is a trial and error method used by application programs to decode encrypted data such as password. It work by calculating every possible combination of password that could make up a password and testing it to see if it is the correct password. As the length of password increases, the amount of time, on average, to find the correct password is also increases.In a brute-force attack repeated credential pairs are tried in an attempt to gain access to an account[2].

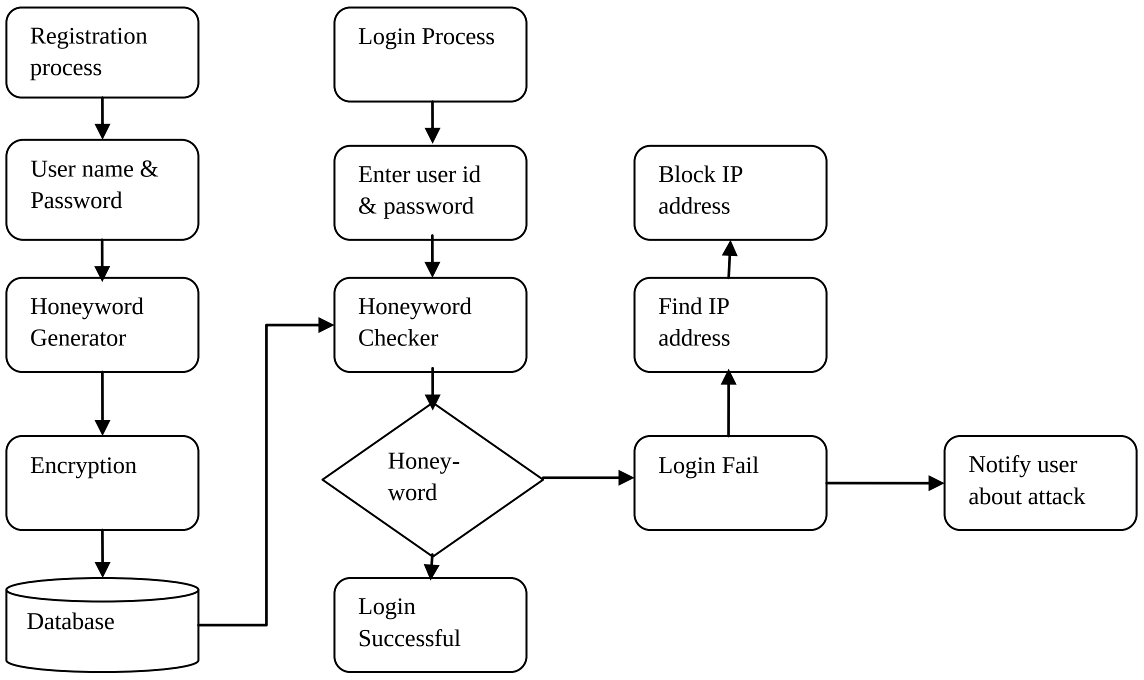

Fig 1: System Architecture

3.2 Initialization

First, fake user accounts (honeypots) are created with their passwords. Also an index value between (1, N), but which is not used previously assigned to each honeypot randomly. Then k -1 numbers are randomly selected from the index list and for each account a honeyindex set is built like Xi ($x_{i,1}$, $x_{i,2}$,. . . , $x_{i,k}$), one of the elements in X_i is the correct index (sugarindex) as C_i[12]. Now, we use two password files as F1 and F2 in the main server: F1 stores username and honeyindex set, <hu_i, Xi> pairs, where hu_i denotes a honeypot account and F2 stores the index number and the corresponding hash of the password, < C_i, $H(P_i)$ > where H(Pi) is hash value of passwords.

3.3 Registration

After the initialization process, system is ready for users registration. A username and password are taken from the user as U_i, P_i to register the system, where $U_{i.}$ is for user id and P_i is for Password. We use the honeyindex generator algorithm Gen (k,S_I) c_i X_i, which outputs c_i as the correct index for u_i and the honeyindexes X_i =($x_{i,1}$, $x_{i,2}$, … ; $x_{i,k}$). Then honeyindex set generation process start with different algorithms, F1 store honeyindex set and F2 store hash passwords corresponding to account.

Last, periodically honeyindexes of each account should be regenerated. As the number of users in the system increases to provide uniform distribution of honeyindexes across S_I, the fresh honeyindex set must involve numbers from this new larger list. Or else, newly created account password would not be used as honeywords in the system and the attacker may get clues to in guessing the correct password for the new user accounts. Note that within a uniform distribution each password is assigned as a honeyword about k times, because there are N passwords but Nk honeywords.

4. HONEYWORD GENRATION METHODS

4.1 Salt Encryption:

A Salt is random data which is used as an additional input to a one-way function that “hashes” data, a password or passphrase in cryptography. To defend against dictionary attacks or against its hashed equivalent, a pre-computed rainbow table attack is the primary function of salts.

Salts are used to safeguard passwords in storage. Previously password were stored in plain text on a system, but recently additional safeguards are used to protect a user's password against being read from the system. A salt is one of those methods.

Example :
Example of a salt value for storing passwords. The first table has two username and password combinations. The password is not stored.

Username	Password
User1	password123
User2	password123

The salt value will be generated randomly and could be any length, in this case the salt value is 8 bytes long. The hashed value is the hash of the salt value appended to the plaintext password. Both the salt and hashed values are stored.

Username	**Salt value**	**String to be hashed**	**Hashed value = SHA256 (Password + Salt value)**
user1	E1F53135E569C253	password123+E1F53135E569C253	72AE25495A7981C40622D49F9A52E4F1565C90F048F59027BD9C8C8900D5C3D8
user2	84B03D034B409E4D	password123+84B03D034B409E4D	B4B6603ABC670967E99C7E7F1389E40CD16E78AD38EB1468EC2AA1E62B8BED3A

From the table above, different values of salt will create completely different hashed values, even when the plaintext passwords are same. Additionally, a dictionary attack is mitigated to a degree as an attacker cannot practically precompute the hashes. A salt cannot protect against common or easily guessed passwords. A complete password storage scheme would also include a salt in it. The first step is to convert the user password in bytes, then the bytes are converted in the hex value with hex encryption technique. The honeyward password will be generated by the combination of salt and hash value.

4.2 SHA 256:

A cryptographic hash is a kind of signature for a data or a text file. SHA-256 generates an unique 256-bit i.e 32-byte signature for a text.

A hash is not an encryption, the original text cannot be decrypted back by hash. This makes it capable when it is appropriate to compare the hashed versions of texts, as opposed to obtain the original text by decrypting the text.

5. HONEY-CHECKER

The honeychecker is executing two commands sent by the main server:

Set: C_i, U_i

Sets correct password index C_i for the user name U_i.

Check: U_i, j

Checks whether C_i for U_i is equal to given value j. Returns the result and if equality does not hold, notifies system a honeyword situation and block IP address.

In the login process the functions of the honeycheker are describe.

6. LOGIN PROCESS

System initially checks whether entered password is correct for the corresponding username U_i. To check this, first the X_i of the corresponding U_i is attained from the F1 file. Then, the hash values stored in F2 file for the respective indices in Xi are compared with hash value of original password to find a match. If a match is not obtained, then it means that password is neither the correct password, nor one of the honeywords then login fails.

If hash value of password is found in the list, then the main server checks whether the account is a honeypot. If it is a honeypot, then it follows a predefined honeypot security policy against the password disclosure scenario. The system provides three attempts to each user.

If hash value of the password is in the list and it is not a honeypot, then honeychecker checks if it is a honeyword or not. If it is a honeyword then system displays the message on the display screen and block IP address and if not then login successfully.

7. CONCLUSION

In this research, we have analyzed the security of the honeyword system. There are several security parameters that are put in place to increase security of password. We have represented the standard approach for securing the system. With the help of analysis author have understood that the strength of the honeyword system is directly depends on the honeyword generation algorithm. Using honey encryption algorithm we can also provide more security for user accounts. Author used Salt and SHA256 for honeyword generation method in the system and also introduce IP address blocking method. If an unauthorized person tries to access the account, the system can automatically generate the alarm or give notification to user. In the honeyword system even if all hashes in the password file has been broken, then also the attacker cannot login to the system without a high risk of being detected.

8. ACKNOWLEDGMENTS

The authors would like to thank the publishers, researchers for making their resources available and teachers for their guidance. We would also thank the college authority for providing the required infrastructure support. Finally, we would like to extend our heartfelt gratitude to our friends and family members.

9. REFERENCES

[1] D. Florencio and C. Herley, 2007, *"A large-scale study of web password habits*", pp. 657–666.

[2] C. Herley and D. Florencio, 2008, ,"*Protection financial institutions from brute-force attacks*", in Proc.23rd Int. Inform. Security Conf., pp. 681-685.

[3] M. Weir, S. Aggarwal, B. de Mederios and B. Glodek, 2009, "*Password cracking using probabilistic context-free grammars*", in Proc. 30th IEEE symp. Security Privacy, pp. 391-405.

[4] P. G. Kelley, S. Komanduri, M. L. Mazurek, R. Shay, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, and J. Lopez, 2012, "*Guess again (and gain and again): Measuring password strength by simulating Password-cracking algorithms*", in Proc. IEEE Symp. Security, Privacy, pp. 523–537.

[5] A. Juels and R. L. Rivests, 2013, "*Honeywords: Making password cracking detectable*", in Proc. ACM SIGSAC Conf. Comput. Commun. Security, pp. 145–160.

[6] K. Brown, 22,Nov.2013, "*The dangers of weak hashes*", SANS Institute InfoSec Reading Room, Maryland US,pp.1–22,Nov.2013 [Online].Available:http://www.sans.org/reading room/whitepapers/authentication/dangers-weak-hashes-34412.

[7] M. H. Almeshekah, E. H. Spafford, and M. J. Atallah, 2013, *"Improving security using deception*", Center for Education and Research Information Assurance and Security, Purdue Univ., West Lafayette, IN, USA: Tech. Rep. CERIAS Tech. Rep. 2013-13, 2013.

[8] Z. A. Genc, S. Kardas, and M. S. Kiraz, 2013 , "*Examination of a new defense mechanism: Honeywords*", IACR Cryptology ePrint Archive, Report 2013/696, 2013.

[9] A. Pathak, , 2014, "*An analysis of various tools, methods and systems to generate fake account for social media*", Ph. D. dissertation, Northeastern University Boston, Boston, MA, USA, 2014.

[10] Imran Ergular, 2016, "*Achiving Flatness: Selecting the Honeywords from Existing User Passwords*", IEEE TRANSACTION ON DEPENDABLE AND SECURE COMPUTING, VOL.13, NO.2, MARCH/APRIL 2016.

[11] Prof. S.P.Khedkar, Bhavana Bachhav, Pratiksha Parsewar, Rakshanda Tirmal, 2016, " *Achieving Flatness by Selecting the Honey words from Existing User Passwords*", International Journal of Engineering Science and Computing, May 2016.