Security Enhancement by Achieving Flatness in Honeywords from existing user passwords

Karishma B. Ajab[1] , Pranali P. Borchate[2], Ashwini A. Jadhav [3], Shubhangi D. Jadhav[4]

.
[1] Karishma B. Ajab, Computer Department, SGOI COE, Maharashtra, India.
[2] Pranali P. Borchate, Computer Department, SGOI COE, Maharashtra, India.
[3] Ashwini A. Jadhav, Computer Department, SGOI COE, Maharashtra, India.
[4] Shubhangi D. Jadhav, Computer Department, SGOI COE, Maharashtra, India

ABSTRACT

Every year new approach against cyber security threat are introduced. But simultaneously the adversary also create new techniques those overcome these efforts. So considering for security and data protection as a priority new technique are needed. So, there is one of the important security issues is with disclosure of password file. To overcome this issue we introduce honeywords (fake passwords) to detect attacks against hashed password databases. For each user account actual password is stored with honeywords. Adversary who steals a file of hashed passwords cannot be sure if it is the actual password or a honeyword for any account. If adversary enters the honeyword for login then it will trigger an alarm notifying the administrator that password file breach. In our system, if the number of attempts exceeds the count of three or entered password other than honeywords then the access will be issued but the files available will be decoy files. Also we suggest an alternative approach to provide realistic honeywords a perfectly flat honeywords generation approach and also to reduce storage cost of the honeywords scheme.

Keywords :- *Honeywords, Honeypot, Authentication, Security, Salting, Password hashing.*

INTRODUCTION

In authentication process it becomes difficult to handle security of passwords that’s why password became the most important asset to authenticate. But users choose the passwords that are easy to remember that can be predicted by the attacker using different attacks like brute force, dictionary, rainbow table attacks etc. So Honeywords plays an important role to defense against stole password files. Specifically, fake passwords placed in the password file of an authentication server.

Generally in many software industries and companies store their data in databases like ORACLE, MySQL or may be other. So, the starting point of a system which is required user name and password are stored in database. Once a password file is bagged, by applying the password cracking technique it is easy to capture most of the plaintext passwords. So for skipping it, there are some issues that should be considered to overcome these security problems:

- First passwords must be safe and secure by using the relevant algorithm.

- Second point is that a secure system should detect the entry of unauthorized user in the system.

1. LITERATURE SURVEY:

1. The most important concept is information security requirement in this which is secured using some authentication method. Various authentication method are existing such as Patterns, Passwords, PIN's etc.. Now-a-days most generally used technic for authentication is passwords. Security of password is an important part in security. A password is a secret word, which a user must input during a login, this word is match only after that it is possible to get access. Generally disclosure of password files is a several security problem that has affected millions of users and many companies and software industries store their data in database, Like Facebook, Yahoo, RockYou, Gmail and Adobe[1],[2].
2. Generally user name and passwords are stored in a database. Since stolen passwords make the users target of many possible attacks. These recent events have proved that the weak password storage methods are currently used by many people on websites. 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[3].
3. Once a password file is leakage, attacker by using the password cracking technique it is easy to capture most of the plaintext passwords[4].
4. In this respect, there are two issues that should be considered to avoid these security problems: First, passwords must be protected by taking proper caution and storing with their hash values computed through some other correct complex mechanisms. Hence, for an advance it must be hard to include hashes value in plain text passwords. The second point is that a secure system should detect whether a password file leakage incident happened or not to take appropriate actions. Honeypot is one of the methods to identify occurrence of a password database breach. In this approach, the administrator purposely creates deceit user accounts to lure adversaries and detects a password disclosure, if any one of the honeypot passwords get used [5], [6].

2. ARCHITECTURE:

2. 1. EXISTING SYSTEM:

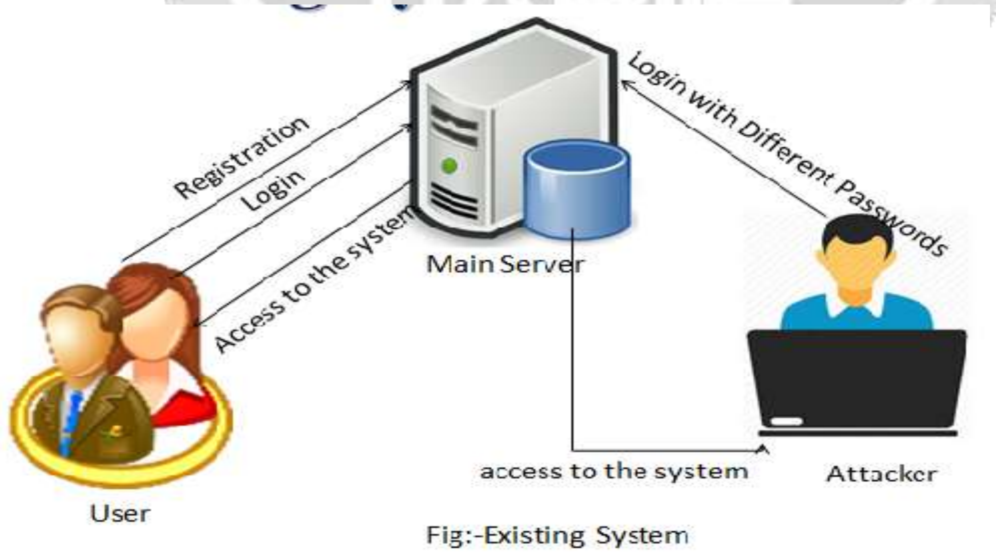

Fig:-Existing System

2.2 PROPOSED SYSTEM:

Our proposed model is still using honeywords concept. However, instead of generating honeywords we generate honeyindexes of existing passwords. For achieving this, for each account we assign index number, which we call honeyindexes. Moreover, hash of the correct password is saving with the correct index in a list. On the other side, in another list u_i is stored with a honeyindex set which is consisting of the honeyindexes and also the correct index. Honeyindex set is created as when any new user registered it takes some honeyindexes and then merge the new honeyindex of that new user and shuffled all the honeyindexes after shuffling we get the new honeyindex set which will get stored in the list. Whenever new user registered all the rows in second list get shuffled. So, when hacker analyses the two lists, he recognizes that each username is matched with k numbers as sweet indexes and each of points to real passwords in the system. The tentative password indexes box an adversary to make a precise guess and he cannot be easily sure about which index are the correct on and other hand shuffling of records and honeyindexes in honeyindex set increases the complexity. The contribution of our approach: First, this model requires less storage compared to the original study. Second, effectiveness of the honeyword system directly depends on how Gen() flatness is creating the honeywords and how it depends on human behavior in choosing passwords. In our approach indexes of passwords of other users are used as the fake indexes in honeyindex set, so it's difficult to find which password is wrong and which is correct becomes more complicated for an adversary.

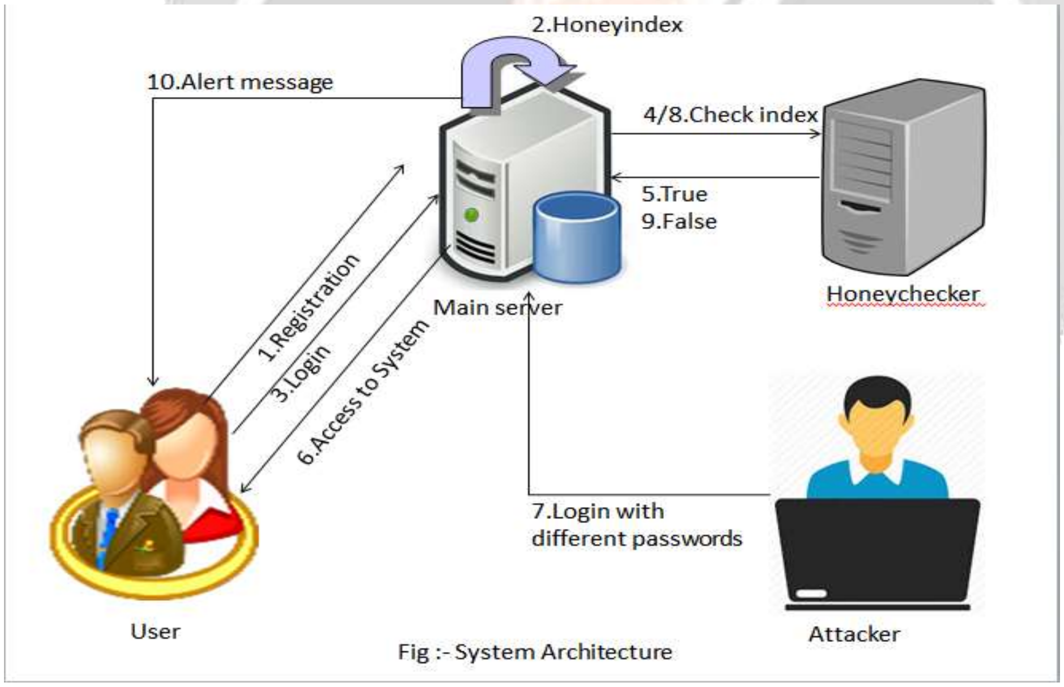

Fig: System Architecture

3. MODULES:

3.1 Initialization:

- This module consists of creating some fake user accounts (Honeypot) are created with their passwords.

- Also an index value between [1, N], but not used previously is assigned to each honeypot at random.

- Then k – 1 numbers are unsystematically selected from the index list Si and for each account a honeyindex set is built like k, S_i =(S_i,1,S_i, 2,...,S_i, k); where c_i is the correct index (sugarindex).

- Two tables are maintained like F1. The first one is the username of the account and the second element is honeyindex set for the respective account. Also, the table is will get shuffled after every new entry.

- On the other hand, F2 keeps the index number and the corresponding hash of the password, <c_i, H(g)> In this case, each entry in the table has two elements. The first aspect is the sugarindex of the account and the second one is the hash of the corresponding password.

User name	Honeyindex set
Lisa134	(93; 16626; : : : ; 94931)
Ravi99	(15476; 51443; : : : ; 88429)
Baba78	(3; 62107; : : : ; 91233)
…	…
…	…
Zoommm0	(1009; 23471; : : : ; 47623)
India67	(63; 51234; : : : ; 72382)

TABLE 1 Example Password File F1 for the Proposed Model

3.2 Registration:

Username and password are required from the user to register the system. Honeyindexes are generated periodically honeyindex set of each account should be regenerated.

3.3 Honeychecker:

Is an auxiliary service, honeychecker is employed to store correct indexes for each account. The honeychecker executes two commands sent by the main server:

1) Set: c_i, u_i

Sets correct password index ci for the user u_i.

2) Verified: u_i, j

Verifies whether ci for u_i is equal to given j. Returns the result and if equality does not hold, notices system a honeyword situation. Thus, the honeychecker only knows the correct index for a username, but not the password or hash of the password.

3.4 Login:

• System firstly checks whether entered password, g, is correct for the corresponding username u_i.

• Then, the hash values stored in f1 file for the respective indices in k, S_i are compared with H(g) to find a match.

• If a match is not found, then it means that g is neither the correct password, nor one of the honeywords, i.e. login fails.

• If H(g) is found in list, then the main server checks whether the account is a honeypot. If it is a honeypot, then the attacker is get redirected to the fake application and log is get created.

• If, however, H(g) is in the list and it is not a honeypot, the corresponding j S_i is delivered to honeychecker with username as $<u_i, j>$ to verify that it is the correct index.

• Honeychecker checks whether $j=c_i$ and returns the result to the main server. At the same time, if it is not equal, then it assured that password is a honeyword and alert is send to the main user about someone is trying to access your account.

3.5 Hacker: Here hacker login to the system. Here if hacker tries to access the system and if he enters any honeyword then the notification or alert message is given to the Actual user.

3.6 Log Creation: Log creation is done for every user actions to the system and which is store into the database.

4 SECURITY ANALYSES:

In this section, we investigate the security of the proposed model against some possible attack scenarios. In Honeywords concept comes with DoS attack sensitivity in which an adversary deliberately tries to login with honeywords to trigger a false alarm.

When a user logins with a wrong password, but not a honeyword, the login fails. If this wrong password is the password of another account in the system and the same user hits this situation more than once (trying with other passwords in F2), the system should turn on additional logging of the user's activities to detect a possible DoS attack and to attribute the adversary, besides the incorrect login attempt case proceeds as usual.

4.1 DOS attack:-

The attacker does not have the password files; his main motive is to prompt a false alarm and to raise a honeyword alarm situation, i.e. depending on the policy some or all parts of the system may be out of service or disabled unnecessarily. We suppose that the attacker knows about m + 1 username and their passwords in the system as (u_a; p_a; : : : ; u_{a+m}; p_{a+m}); maybe he intentionally made all of these accounts. Here, a possible method is creating m accounts with the same password as p_z, while a single account u_y, has different password like p_y and entering the system with the username u_y and the password p_z. If p_z is assigned by the system as a honeyword, then the attacker mounts a DoS attack by entering with the system $< u_y, p_z >$ pair. Let $P_r(p_z / W_y)$ denote the probability that p_z is allocate as one of the honeywords for u_y; it is also the success probability of the attacker for this attack.

4.2 Password Guessing:-

In this attack, we assume that the attacker has stolen password files F1 and F2 from the main server and it's not possible to invert the hash value created by SHA2 algorithm but still we assume that attacker somehow manage to obtained plaintext passwords by inverting the hash values. Extracted F2file (after inverting hashes) gives < index number, password > pairs to the attacker, but these are not directly connected to a specific username. By just analyzing this, he cannot exactly determine which password belongs to which user. On the other hand, F1 gives username; index set pairs such that for each username k possible passwords exist which are shuffled by that it becomes more complicated to find the correct password. If the attacker tries to compare both the files F1 and F2 and tries to guess the password, it's not possible because after every new entry in file F1 shuffle the rows. Attacker has no advantage in guessing the correct password by using specific information about the user, such as age, gender and nationality. If the attacker randomly picks an account from the list in F1 and then tries to login with a guessed password, then her success will depend on: First, the selected account is not a honeypot (decoy) account. Second guessing the correct password p_i out of k sweetwords. Otherwise, the attacker will be caught by the system due to a honeyword or a honeypot.

4.3 Brute-force Attack:-

We suppose that if a honeypot entrance is detected by the system, it responds with a strong reaction, while a light policy (not suggested) is executed in case of honeyword detection. So, we assume that even in honeyword detection the attacker may proceed to make her trials due to light local policies. If, however, a honeypot account is attempted then system follows a strong policy e.g. demanding all users to renew their passwords. From binomial distribution the probability that the attacker hits at least one honeypot in his trials. It is equivalent to say that in brute- force guess attack, it is likely that the attacker hits a honeypot and system detects the password disclosure situation.

5. CONCLUSION:

In this research, we have analyzed the security of the honeyword system and addressed a number of flaws that need to be managed before successful realization of the scheme. The strength of the honeyword system directly depends on the generation algorithm, i.e. flatness of the generator algorithm determines the chance of distinguishing the correct password out of respective sweetwords. Another thing that we would like to stress is that defined reaction policies in case of a honeywords entrance can be exploited by an adversary to realize a DoS attack. This will be a serious threat if the chance of an adversary in hitting a honeyword given the respective password is not negligible. We have noted that the security approach should strike a balance between DoS vulnerability and effectiveness of honeywords. As per the Imran Erguler, chaffing-by-tweaking method is weak and also some weaknesses of the chaffing-by-tweaking techniques are accepted by their creators. Moreover, the chaffing- with tough nuts model has been investigated, and we have doubted about its favor as opposed to ideas of Juels and Rivest. Finally, we come up with a new approach to make the generation algorithm as close as to human nature by creating honeywords with randomly picking passwords that belong to other users in the system. In this we use SHA2 which becomes difficult to invert the hash values into the passwords as well as we shuffle the records in files and also the honeyindexes in the honeyindex set which makes guessing of correct password more complicated.

6. REFERENCES:

[1]. D. Mirante and C. Justin, "Understanding Password Database Compromises," Dept. of Computer Science and Engineering Poly-technic Inst. of NYU, Tech. Rep. TR-CSE-2013-02, 2013.

[2]. A. Vance, "If Your Password is 123456, Just Make It Hackme," The New York Times, vol. 20, 2010.

[3]. K. Brown, "The Dangers of Weak Hashes," SANS Institute InfoSec Reading Room, Tech. Rep., 2013.

[4]. M. Weir, S. Aggarwal, B. de Medeiros, and B. Glodek, "Password Cracking Using Probabilistic Context-Free Grammars," in Security and Privacy, 30th IEEE Symposium on. IEEE, 2009, pp. 391–405.

[5]. F. Cohen, “The Use of Deception Techniques: Honeypots and Decoys,” Handbook of Information Security, vol. 3, pp. 646–655, 2006.

[6]. M. H. Almeshekah, E. H. Spafford, and M. J. Atallah, “Improving Security using Deception,” Center for Education and Research Information Assurance and Security, Purdue University, Tech. Rep. CERIAS Tech Report 2013-13, 2013.

[7]. J. A and L. R. R, "Honeywords: Making Password cracking Detectable," in ACM SIGSAC conference on Computer & communications security, November 2013.

[8]. Imran Erguler, "Achieving Flatness: Selecting the Honeywords from Existing User Passwords," IEEE Transactions on Dependable and Secure Computing, vol. 13, no. 2, pp. 284 - 295, February 2015.

[9]. Z. A. Genc, S. Kardas, and K. M. Sabir, “Examination of a New Defense Mechanism: Honeywords,” Cryptology ePrint Archive, Report 2013/696, 2013.

[10]. J. Bonneau, “The science of guessing: Analyzing an anonymized corpus of 70 million passwords,” in Security and Privacy (SP), 2012 IEEE Symposium on. IEEE, 2012, pp. 538–552.