RESEARCH ARTICLE OPEN ACCESS

Detecting and Analyzing Password Database crack Using Honeyindex

[1]Priyanka Anil Ghule, [2]Prof. Sonali Bhutad
[1,2]*Computer Engineering, Shah & Anchor Kutchhi Engineering College, Mumbai*

----------**********----------

Abstract:

In the honeywords approach default passwords are used to detect attacks against hashed password databases. For each user account, the list of password is stored with several honeywords. If honeywords are selected properly, an attacker who steals a file of hashed passwords cannot be sure if it is the real password or a honeyword for any account. Login with a honeyword will trigger an alarm notifying the administrator about a password file breach. At the expense of increasing the storage requirement, the honeyindex system is an effective solution to the detection of password file disclosure events. Also, as this approach uses existing user index as a honeyindex of other user it reduce storage cost as compare to honeyword scheme.

Keywords —- **Honeyindex, honeywords, passwords, password cracking, chaffing, login, authentication**.

----------**********----------

I. INTRODUCTION

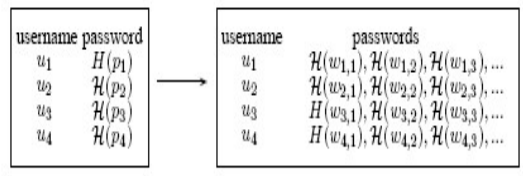
For every web application, in the authentication process, password became the most important asset to login. But users choose weak passwords that can be predicted by the attacker using brute force, dictionary, rainbow table attacks etc. An adversary who has stolen a file of hashed passwords can often use brute-force search to find a password p whose hash value H(p) equals the hash value stored for a given user's password, thus allowing the adversary to impersonate the user. For authentication process one of the most common tool is password. In registration process, most of the users choose weak passwords that can be predicted by a brute-force attack. An adversary, who steals the file of hashed passwords from a server, can use brute force attack to find some user's password. Weir et al. [16] developed a password cracking algorithm which uses probabilistic, context-free grammars.

So Honeywords are a defence against stolen password files. Specifically, they are bogus passwords placed in the password file of an authentication server to deceive attackers. It's hard therefore for an attacker that steals a honeyword password file to distinguish between honeywords and true user passwords.

Hackers who steal databases of user logins and passwords only have to guess a single correct password in order to get access to the data. Hackers knows the correct password through the database or when file becomes readable.

II.TECHNICAL DESCRIPTION

In [9], Juels and Rivest recently propose the idea of changing the structure of the password file in such a way that each user would have multiple possible passwords, *sweetwords* and only one of them is real. The false passwords are called *honeywords*. As soon as one of the honeywords is submitted in the login process, the adversary will be detected. Let ui, pi and H() denotes the ith user name, her password and the hash function of the standard system respectively. As in Figure 1, the system adds honeywords hashes to this file at random orders. Thus an adversary who has cracked the password hashes will see randomly ordered sweetwords wi,j of user ui.



When a user ui sends a login request, the login server will determine the order of her among the users, and the order of the submitted password among her sweetwords. The login server sends a message of the form Check (i, j) to a secure server which is called "honeychecker", for the ith user and her jth sweetword.

A closely related work is the Kamouflage system of Bojinov et al. [3] though that work differs from honeywords. In that system, the user's password list is placed with another list that contains honeywords. There is no need for a server in Kamouflage system although the authors in [3].

A. Attack models

There are numerous attacks to obtain a user's password. The six of these techniques are depicted in Figure 2.

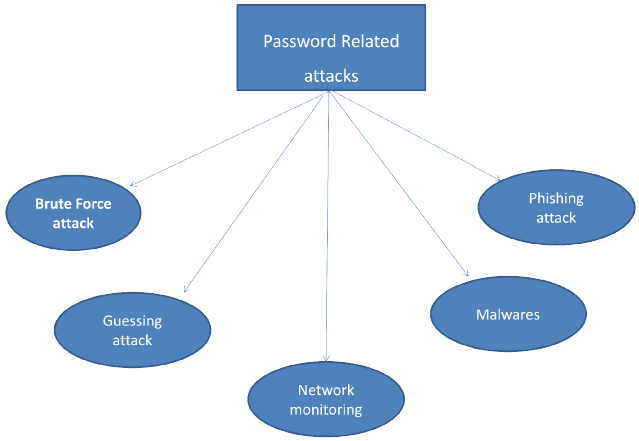


Fig. 2. Password Related Attacks

Password attacks can be classified as follows:

• **Brute-force attack:** An adversary can steal the password hash file and crack the hashes using brute force computation and also use a precomputed dictionary of password hashes [6].

• **Guessing attack:** Many users choose weak passwords such that an adversary can find out the passwords of some users of a system by trying common passwords while attempting to login to that system [4], [5].

• **Network monitoring:** If the communication between the user and the system is unsecured, i.e. unencrypted, an adversary may monitor the network traffic and obtain the passwords or interrupt the traffic while a user creating her password and change it to another one [12]. This attack is also called man-in-the-middle-attack.

• **Phishing attack:** A user can submit her login information to a web page prepared by an adversary which seems very likely to the original system's login screen.

• **Malwares:** There are some advanced malwares that can steal the login information from messenger like softwares some of which does not keep the login information encrypted [8]. A Trojan program can capture the key strokes and send this information to the adversary [7].

B. Honeychecker

honeychecker is an auxiliary secure server to work with the honeywords. The honeychecker is a separate computer system where such secret information can be stored. The computer system can communicate with the honeychecker when a login attempt is made on the computer system, or when a user changes her password and also assume that the honeychecker is capable of raising an alarm when an irregularity is detected. The alarm signal may be sent to an administrator or other party different than the computer system itself. The honeychecker accepts commands of exactly two types:

• Set: i, j
Sets c(i) to have value j.

• Check: i, j
Checks that c(i) = j.
returns result of check to requesting computer system and raise an alarm if check fails.

C.Honeyword Generation Methods

There are several methods of generating honeywords as follows.

1.Chaffing-by-tweaking:

In this method, the user password seeds the generator algorithm which tweaks selected character positions of the real password to produce the honeywords. Each character of a user password in predetermined positions is replaced by a randomly chosen character of the same type: letters by letters, digits are replaced by digits, and special characters by special characters. Number of positions to be tweaked, denoted as t should depend on system policy. As an example t = 3 and tweaking last t characters may be a method for the generator algorithm Gen(k; t). Another approach named in the study as "chaffing-by-tweaking-digits" is executed by tweaking the last t positions that contain digits.

42hello and t = 2, the honeywords 12hello and 58hello may be generated.

2. Chaffing-with-a-password-model:

This method generates honeywords using a probabilistic model of real passwords; this model might be based on a given list L of thousands or millions of passwords and perhaps some other parameters. Unlike the previous chaffing methods, this method does not necessarily need the password in order to generate the honeywords, and it can generate honeywords of widely varying strength. Here is a list of 19 honeywords generated by one simple model :

ksdebrton1	0653487dia
ads41ger	forlindsdux
1erapc	sbgo864959
aiwkme5323	aj1aodasb12
9,50PEes]KV.0?RIOtc&L-:IJ"b+Wol<*[!NWT/pb	
xyqi3tbadto	fa3915
@#NDYRODD_!!	Ffvenlorhan
dspizzhemix01	dfffdhusZ2
dfssveniresly	'S43b123
sdmobopy	WORFmgthness

III A NEW APPROACH

Proposed model is still based on use of honey words to detect password-cracking. Here instead of generating the honeywords and storing them in the password file, this approach suggested benefiting from existing passwords to simulate honeywords. In order to achieve this, for each account k - 1 existing password indexes, which we call honeyindexes, are randomly assigned to a newly created account of ui, where k >= 2. Moreover, a random index number is given to this account and hash of the correct password is kept with the correct index in a list. On the other hand, in another list ui is stored with an integer set which is consisted of the honey indexes and the correct index. So, when an adversary analyses the two lists, she recognizes that each username is paired with k numbers as sweet indexes and each of which points to real passwords in the system. The tentative password indexes hamper an adversary to make a correct guess and he cannot be easily sure about which index is the correct one. It is equivalent to say that to create uncertainty about the correct password, system propose to use indexes that map to valid passwords in the system. This method requires less storage compared to the original study. Within this approach passwords of other users are used as the fake passwords, so guess of which password is fake and which is correct becomes more complicated for an attacker.

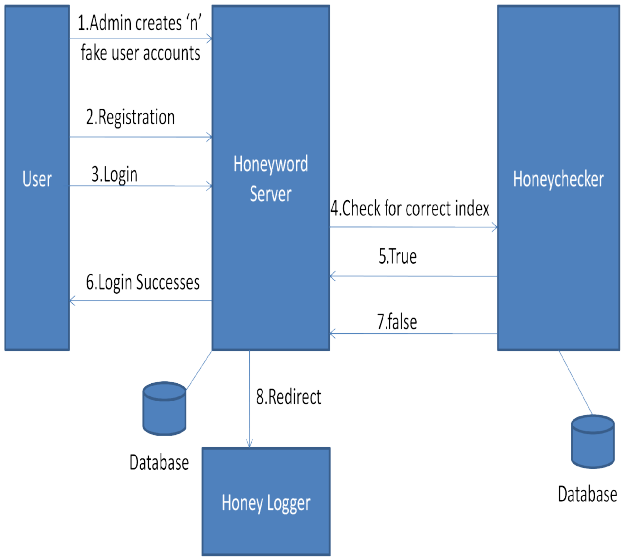


Fig3. System Architecture

A.Modules

1. *Initialization:*

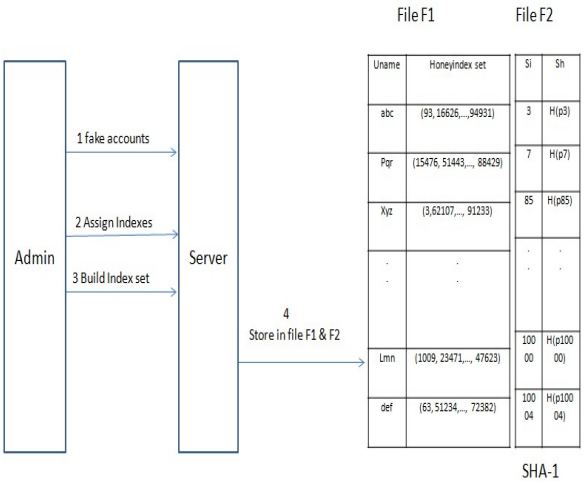


Fig 4 Initialization

- This module creates some fake user accounts (Honeypot).
- Also an index value which is unique is assigned to each honeypot randomly.
- Then k – 1numbers are randomly selected from the index list and for each account a honeyindex set is built like Xi = (xi,1,xi,2,...,xi,k); one of the elements in Xi is the correct index (sugarindex) as ci.
- Two tables are maintained first is F1 which stores username and honeyindex set (the table is sorted alphabetically by the username field) and second is F2 which stores the correct index number and corresponding hash of the password (the table is sorted according to the index values).

***2.Registration*:**

- In this module first user register by adding his personal information like his name, password, address, Contact number, email etc.
- After registering with the application unique honeyindex is get randomly assigned to that user and hash of the password is get stored in file F2. The honeyindex set is built and name, honeyindex set is getting stored in file F1.
- Periodically honeyindex set should be regenerated.

3. Honeychecker:

- An auxiliary service honeychecker is used to store correct indexes for each account.
- Honeychecker can be anything background service or isolated DB or a server at remote secure location.
- Requests, replies and updates sent to the honeycheckers need to be authenticated.
- If the communications between computer system and honeychecker is disabled by adversary then system goes into failover mode.The honeychecker executes two commands sent by the main server:

1) Set: ci,ui Sets correct password index ci for the user ui.
2) Check: ui,j Checks whether ci for ui is equal to given 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. [8]

Login:

System checks whether entered password g, is correct for the corresponding username ui.To accomplish this, firstly the Xi of the corresponding ui is attained from the F1 file.

- Then, the hash values stored in F2 file for the respective indices in Xi are compared with H (g) to find a match.
- If a match is not obtained, 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 the 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.
- If however, H (g) is in the list and it is not a honeypot, the corresponding j∈ Xi is delivered to honeychecker with username as <ui,j> to verify it is the correct index.
- Honeychecker checks whether j=ci and returns the result to the main server. At the same time, if it is not equal, then it assured that the entered password is a honeyword

and alert is send to the main user about someone is trying to access your account.

- There are various types of logins such as –

- **Admin login**
  Here admin can login into the system. Admin user is responsible for creating fake accounts. Once login he can handle all administrative functions.
- **User login**
  Here user is going to login into the System. If password matches with the hash password then user can login**.**
- **Hacker login**
  Here hacker is going to login the system. Here if hacker tries to break the system and if he enters any honeyword then the alert is given to the actual user. If hacker accesses the honeypot account then he will get redirected towards the fake application.
- **Log Creation**
  Log creation is done for user those who are directed towards honeylogger.

IV SECURITY ANALYSIS OF THE PROPOSED MODEL

- In this section, we investigate the security of the proposed model against some possible attack scenarios. Some set of reasonable assumptions about our approach and the related security policies. We suppose that the adversary can invert most or many of the password hashes in file F2.
- When a user logins with a wrong password, but not a honey word, 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.
- If a password, whose hash value is entered in wrong login attempts for more than once, the system should take actions against a possible DoS alarm. In this case the system suspects about the respective password such that it is known by the adversary (possibly she created an account with this password) and she aims to raise a honeyword situation. Resultantly, the consecutive wrong login attempts with this password gives rise to a DoS warning and further activities of the user are investigated by the admin as a precaution to prevent a false honeyword alarm.

A. DoS Attack

Hacker's main purpose is to trigger 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 adversary has knowledge m + 1 username and respective passwords in the system as (ua; pa; : : : ; ua+m; pa+m); maybe she intentionally created all of these accounts. In this case, a method for attacking the system is creating m accounts with the same password as Pz, while a single account, Uy, has different password like Py and entering the system with the username uy and the password Pz. If Pz is assigned by the system as a honeyword, then the adversary mounts a DoS attack by entering with the system < uy; pz > pair.

B. Password Guessing

In this attack, we assume that the adversary has plundered password files F1 and F2 from the main server and also obtained plaintext passwords by inverting the hash values. Extracted F2 file (after inverting hashes) gives < index number; password > pairs to the adversary, but they are not directly connected to a specific username. By just analysing this, she 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. Also, we suppose that the adversary has no advantage in guessing the correct password by using specific information about the user, such as age, gender and nationality. If the adversary 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 pi out of k sweetwords. Otherwise, the adversary will be caught by the system due to a honeyword or a honeypot.

V.COMPARISON OF HONEYWORD GENERATION MODELS

In this section, we give a comparison of the generation methods including our proposed model with respect to storage cost, DoS resistance of each algorithm. N operations must be executed. On the other hand, according to the Juels and Rivest's method she needs to launch k operations for a specific user and kN operations for the whole space, since for each user k sweetwords are assigned. Thus, one can see that the adversary has to spend k times more effort for each case in this method. Now, for our proposed model, if the attacker focuses on a specific user, she must still try k hash inversions to reveal all possible passwords for this target user. However, since each honeyword is indeed password of another user, revealing all passwords in the system requires N operations as in the case of the traditional system. Therefore, taking the total password-cracking cost of the adversary for the whole system into account, we can say that existing system requires higher effort for an hacker in retrieving plaintext passwords.

A. Storage Cost

Here storage requirement of our method and compare it with that in [9]. A typical password file system requires hN plus storage for usernames, where N stands for the number of users in the system and h denotes length of password hash in bytes. On the other hand this is khN for [9], where k denotes the number of the sweetwords assigned to each account.

In proposed system a storage optimization technique for the chaffing-by-tweaking model such that keeping only hash of a single sweetword, in database would be enough, because the main server can compute all possible honeywords from an entered proffered password g.

. For our approach we assume that each index requires 4 bytes and the storage cost becomes:

$4kN + hN + 4N$

To measure the gain in storage compared to original method, we give the ratio as:

$$\frac{4kN + hN + 4N}{KhN} = \frac{4k+h+4}{kh}$$

Also, as k increases storage cost of our scheme is affected by the term 4kN, while this is hkN for the methods of [9].

B. DoS Resistance

The chaffing-with-tweaking model may suffer from a DoS attack, due to predictability of the honeywords. Unlikely, the chaffing-with-a-password model provides resistance against such an attack, because honeywords are generated by using a list of passwords such that they may be independent from the correct password. Note that the authors in [9] avoid direct use of a password list to eliminate a DoS attack threat in case of very common passwords exist in the list. As opposed to this idea, our proposed scheme uses password list in the system as honeywords of a user. Thus, use of real passwords as honeywords does not cause a DoS weakness.In our proposed model in addition to honeywords, are employed to detect a password disclosure. This facilitates showing a strong response to actions of an adversary, because entering a honeypot account with one of its sweetwords ensures occurrence of a password leakage.

Method	DoS resistance	Storage Cost
Tweaking	weak	hN*
Password Model	strong	KhN
Our Model	strong	4kN+hN+4N

TABLE 1.Comparison of Honeyword Generator Models

VI .CONCLUSIONS

In this study, we have analysed the security of the honeyword system and addressed a number of flaws that need to be handled before successful realization of the scheme. Also defined reaction policies in case of a honeyword entrance can be exploited by an adversary to realize a DoS attack. It has been shown that DoS resistance of the chaffing-by-tweaking method is weak. On the other hand, the chaffing-with-a-password model can fulfill its claims provided that the generator algorithm is flat.

We have compared the proposed model with other methods with respect to DoS resistance, and storage cost and usability properties. The comparisons have indicated that our scheme has advantages over the chaffing-with-a-password model in terms of storage and usability.

REFERENCES

[1] D. Mirante and C. Justin, "Understanding Password Database Compromises," Dept. of Computer Science and Engineering Polytechnic 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 ofWeak 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] C. Herley and D. Florencio, "Protecting financial institutions from brute-force attacks," in SEC'08, 2008, pp. 681–685.

[8] H. Bojinov, E. Bursztein, X. Boyen, and D. Boneh, "Kamouflage: Loss-resistant Password Management," in Computer Security– ESORICS 2010. Springer, 2010, pp. 286–302.

[9] A. Juels and R. L. Rivest, "Honeywords: Making Passwordcracking Detectable," in Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, ser. CCS '13. New York, NY, USA: ACM, 2013, pp. 145–160. [Online].
Available: http://doi.acm.org/10.1145/2508859.2516671

[10] M. Burnett, "The Pathetic Reality of Adobe Password Hints,"https://xato.net/windows-security/adobe-password-hints.

[11] 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.

[12] D. Malone and K. Maher, "Investigating the Distribution of Password Choices," in Proceedings of the 21st International Conference on World Wide Web, ser. WWW '12. New York, NY, USA: ACM, 2012, pp. 301–310. [Online]. Available: http://doi.acm.org/10.1145/2187836.2187878

[13] M. Burnett, "10000 Top Passwords," https://xato.net/passwords/more-top-worst-passwords/.

[14] L. V. Ahn, M. Blum, N. J. Hopper, and J. Langford, "CAPTCHA: Using Hard AI Problems for Security," in Proceedings of the 22nd International Conference on Theory and Applications of Cryptographic Techniques– EUROCRYPT'03, ser. Lecture Notes in Computer Science, vol. 2656. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 294–311.

[15] L. Zhao and M. Mannan, "Explicit Authentication Response Considered Harmful," in Proceedings of the 2013 Workshop on New Security Paradigms Workshop–NSPW '13. New York, NY, USA: ACM, 2013, pp. 77–86. [Online]. Available: http://doi.acm.org/10.1145/2535813.2535822

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

[17] P. G. Kelley, S. Komanduri, M. L. Mazurek, R. Shay, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, and J. Lopez, "Guess again (and gain and again): Measuring Password Strength by Simulating Password-cracking Algorithms," in Security and Privacy (SP), 2012 IEEE Symposium on. IEEE, 2012, pp. 523–537.

[18] J. Bonneau and S. Preibusch, "The Password Thicket: Technical and Market Failures in Human Authentication on the Web," in WEIS, 2010.