Preventing The Exploitation Of Stolen Credentials: Using Non-Realistic Honeywords

Akshaya K[1], Dr. Subramanium Dhanabal[2]
[1]M.Tech. Scholar, Dept. of CSE, NCERC, Kerala
[2]Associate Professor Scholar, Dept. of CSE, NCERC, Kerala

Abstract - We propose a method for preventing the usage of hashed password by the attacker, if password file disclosure happened, using non-realistic honeywords (decoy passwords). Recently, Imran Erguler proposed a flat honeyword generation method which selects the honeywords from existing user passwords of the system in order to provide realistic decoy passwords and for making the stealer confused. By considering the ethic in using someone's credentials, here we are accepting graphical passwords(images), taking a string from the textual form of the image as password and keeping it with a set of unrealistic honeywords with necessary encryption. So hacker who steals the hashed password file cannot distinguish between the real password and the honeywords for any account, since every passwords seem unrealistic. Moreover using a honeyword to login will trigger an alarm notifying the administrator of the system about the password file disclosure incident.

Keywords: Authentication, Honeywords, Login, Password, Password cracking.

I. INTRODUCTION

One of the most common and familiar means of security in online system as well as in real life is by use of password information. A user or an agent requesting access to a restricted resource need to provide a password, and anyone able to enter the right password is considered to be authorized to access the resource. So what if the Password file itself has disclosed?

There are two security issues needs to be considered by every system whose security is by means of passwords[1]. First, there must be a strong password protection mechanism which will take appropriate precautions and store passwords with their hash values computed through some complex mechanisms. Hence the attacker will find it difficulty in inverting the hashes to acquire the plaintext passwords. Second, the security system also needs to detect whether the password disclosure incident happened or not to take appropriate actions.

Some similar ideas have arisen in the literature. Bojinov. et al. in introduced a theft resistant built using decoys. In his model, the fake password sets are stored along with the legitimate password set to conceal them, called as Kamouflage[6] . According to him, an attacker who steels the laptop or cellphone having a Kamouflage-based password manager is forced to carry out a considerable amount of on-line work before obtaining any user credentials. That is, his main aim was to make it harder for the attacker to exploit the stolen password manager. To the best of our belief, the term honeyword first appeared in that work. A patent application by Rao describes the use of decoy passwords per-account called failwords [7] used to trap an adversary into believing he has logged into successfully, when he hasnt.

For developing a system that generate honeywords that are indistinguishable from the correct passwords, we propose a new approach that request an image from user then a string of the textual form of that image will be stored as the password after hashing it. Honeywords are generated randomly , i.e.both the password and the honeywords become unrealistic and indistinguishable.

The rest of this paper is organized as follows. In Section 2, we review the various password and server attacks and its effects. Section 3 discuss about the password disclosure detection approaches, includes honeypots and honeywords and Section 4 describes the identified limitations of the existing password disclosure. Section 5 gives the description of our proposed model. Section 6 will describes the security analysis of the proposed model and section 7 about the significance of the proposed model with experimental evaluation. Section 8 will conclude this paper and section 9 talks about the future scope of our study.

II. LITERATURE SURVEY

To protect ourself from password and server attack, we should become familiar with the most commonly used types of attacks.

- Password Guessing
  Password guessing is one of the most common type of attack. Attackers guess passwords either locally or remotely. Password guessing is not difficult as we would expect. Not all authentication protocols are equally effective against guessing attacks. Password guessing attacks can be of two types:

  Brute Force Attack - In Brute force type of password guessing attack, the attacker will try every possible combination, code and password to find the correct one. This type of attack may take long time to identify the password if it is a complex one.

Dictionary Attack - Here, the attacker use a dictionary of common words to identify the correct password.

- Phishing
  This is an attempt to obtain the sensitive information by disguising as a trustworthy object often for malicious reasons. The disguising entity will try to be look and feel like the legitimate entity and is often directs the users to enter the personal information for their own sake.

- Password Resetting
  Resetting the password is much easier for the attackers than guessing them. Most of the password cracking programs are really password resetters. An attacker need not to get the password from the user if he can make the authentication system to either mail it to him or change it according to his choice.

- Network Monitoring
  The adversary may monitor the traffic and obtain the user credentials if the communication between the user and the system is unsecured. this attack is also known as man-in-the-middle-attack.

- Malwares
  There are some malwares which are able to steal the login information of the system which does not keep the login information encrypted. Trojan is an example for such type of malwares.

- Visible attacks
  This attack is also known as shoulder surfing. He can watch a user while she enters her password.

- Pharming
  In this type of attack, the attacker compromises on the user computer or the Domain Name System (DNS) servers so that traffic is redirected to a malicious site.

- Defacement
  In this type of attack the attacker replaces the organization’s webpage with a different web page with his name and image.

A. Effects Of Successful Attacks

- If the attacker includes links to porn websites, malicious information, or even edited any of the information of the website, then the organization’s reputation can be ruined.
- The attacker may use the webserver to install malicious software.
- The attacker can use the user data for fraudulent activities, and this may lead to business loss or lawsuits from entrusted users of the organization.

Password file disclosure has affected many users and companies like Yahoo, RockYou, LinkedIn, eHarmony and Adobe. These are due to the weak password protection mechanisms without a salt. Salt is the random character set we mixed with the passwords or confidential information before performing hashing[9]. Indeed once a password file disclosure happened, with the help of password cracking techniques it is easy to capture most of the plaintext passwords. Because of the sheer volume of the passwords along with its hints dumped in the same site, about 150 million Adobe passwords leaked[4][5]. Among these large percentage of the the passwords were the name of a person, the name of a pet, the name of a place, or an important date. No matter how many times the administrators of most of the application suggest us not to use the same password across multiple websites, we will do the same because of the difficulty in memorising too many passwords.

Thus along with a strong password storage and password protection mechanism, we also need a security system that could detect whether the password disclosure happened or not to take appropriate actions. Honeywords and honeypots are the two mechanisms for finding the password disclosure. Next section will briefly summarize about these.

III. PASSWORD DISCLOSURE DETECTION APPROACHES

In this section, we discuss about the two approaches which are most commonly used for detecting the password disclosure.

A. Honeypots

Practical but more limitting definition of honeypot is given by pcmag.com. A server that is configured to detect an intruder by mirroring a real production system. It appears as an ordinary server doing work, but all the data and transactions are phony. Located either in or outside the firewall, the honeypot is used to learn about an intruder’s techniques as well as determine vulnerabilities in the real system. Honeypots are broadly categorized in to high interaction and low interaction based on the interaction level, or services provided by the honeypot hackers. Although honeypots have advantages like they can work in encrypted environment, can collect smaller, higher-value, datasets since they only log illegitimate activity, and do not require known attack signatures, they can potentially be detected by the attacker.

B. Honeywords

Honeywords are false passwords, which are stored along with the legitimate password for each user account in order to sense impersonation. Even if the adversary gets the password list and recovers a number of password candidates for each account, she cannot be sure about which word is genuine. Also the cracked password files can be identified by the system administrator if a login attempt is done with a honeyword by the adversary.

We need to be familiar with some terms used in this work. The set of fake passwords added with each correct password is called sweetwords. The sweetwords together with the correct password is called honeywords. The correct password among them is called the sugarword. The user does not need to be panicked by hearing about honeywords. The user only needs to remember her password, she does not need to know the values of the honeywords, or even know about their existence. The coming sections discuss about the systems implemented based on these approaches.

1) *Kamouflage:* For building a theft-resistant password manager Hristove Bojinov developed an architecture and he named it Kamuflage[6]. the main aim of this system is to force the attacker to carry out a large amount of online work before obtaining any of the user credentials. While the standard password managers store a single set of user passwords, this system stores the real set of password with N-1 decoy sets as shown in Fig1. The main challenge is to generate the decoy sets indistinguishable from the real set. With Kamouflage, before recovering the user credentials, the attacker who steals the the device with Kamouflage will be force to perform an average, $N/2$ login attempts. Websites can detect these failed login attempts and act accordingly.

2) *Juels-Rivest:* They[2] have used Gen(k) as the honeyword generation algorithm for generating the sweetword list w_i for each user account u_i, where k is the number of sweetwords to be generated (by default k=20). The algorithm outputs both the password list w_i and c_i, where c_i is the index of the sugarword. The username and the hashes of the honeywords are stored in the main server and c_i is stored in the honeychecker. The auxillary service honeychecker communicate with the main server through authentication channel. The honeycecker only knows the correct index for a username, but not the password or the hash of a password. The login procedures (Fig2) is summarized below.

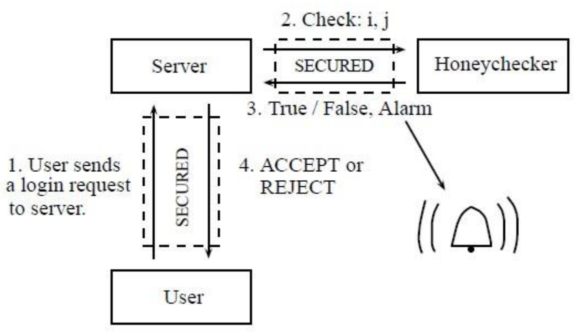


Figure 2. Login schema of a system using honeywords (Z.A. Genc et al., 2013, p. 2).

- User enters a password to login to the system
- The administrator checks whether the hashed string is available in the set of honeywords stored for the corresponding user. If not login denied
- Otherwise the system checks whether it is a sweetword or the sugarword
- If it the correct password, then the system returns a TRUE value, otherwise returns a FALSE value and may also raise an alarm informing the administrator about the password file breach.

3) *Modified Juels-Rivest Model:* In this system of enhanced honeywords, they have suggested some solutions to make the system proposed in[2] more robust. The solutions include: Instead of having a constant number of honeywords per-user as in [2], the number of

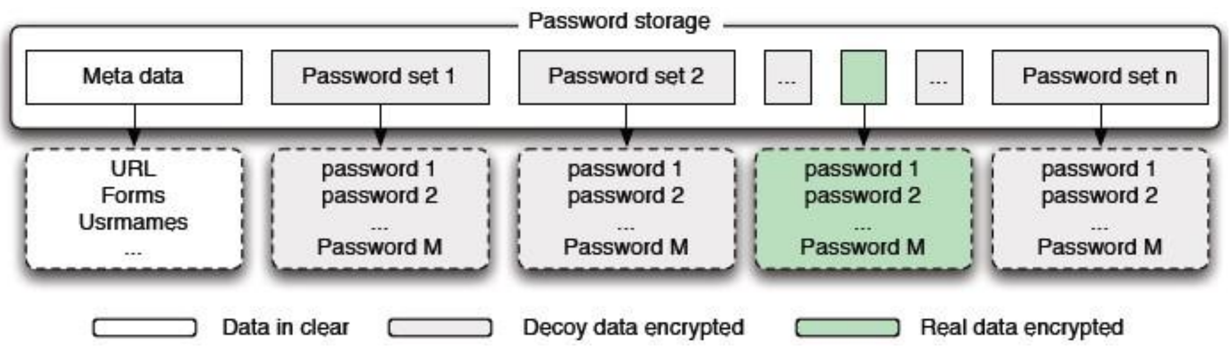


Figure 1. Kamouage database: cleartext metadata and encrypted real and decoy password sets (Bojinov et al., 2010, p. 8).

honeywords for each user should be dynamically chosen by the system[8].

Here the user needs to add an extra parameter *phn*, the user phone number, when he signs up. The honeychecker will update its database with this information and communicate with the user when needed. The sign up scheme of the enhanced honeyword system is depicted in Fig3. If a user would like

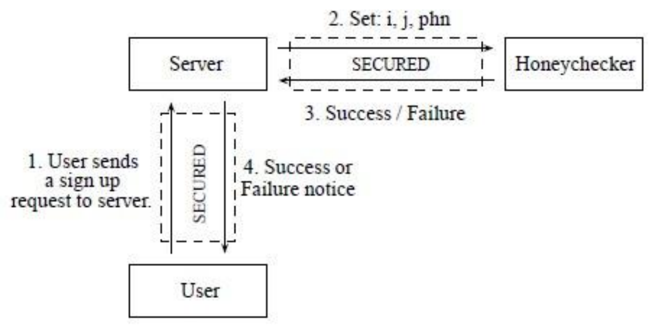


Figure 3. Sign up schema of an enhanced honeywords system (Z.A. Genc et al., 2013, p. 5).

to change his password, he need to send a request to the login server. The login server will send the message to honeychecker, then the honeychecker will communicate with the user with his mobile number to validate the origin of the request. The honeychecker will set an alarm for invalid requests. The password change scheme of an enhanced honeyword system is depicted in Fig4.

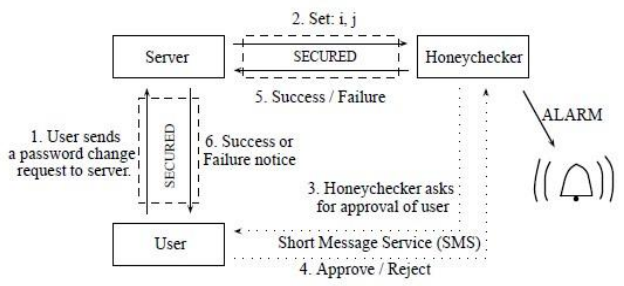

2. Set: i, j
Server
SECURED
Honeychecker
5. Success / Failure
ALARM
1. User sends a password change request to server.
SECURED
6. Success or Failure notice
3. Honeychecker asks for approval of user
User
Short Message Service (SMS)
4. Approve / Reject

Figure 4. Password change schema of an enhanced honeywords system (Z.A. Genc et al., 2013, p. 5).

4) *Erguler Approach:* In this model[1] they have taken the benefit from existing passwords to stimulate honeywords. While creating a new account $k - 1$ honeyindexes are randomly assigned, where $k \geq 1$. The fakeindexes stored for each account is called sweetindexes and the correct index among them is called sugarindex. The u_i and the honeyindex set is stored n one list and the userid and the correct index is stored in another list. So even if the adversary analyses both the lists, he recognizes that each account is paired with k number of sweetindexes each of which points to correct password in the system and cannot be easily sure about which index is the correct one. The algorithm used to generate is Gen(k,S_I) $\rightarrow$ c_i,X_i which outputs c_i as the correct index for u_i and the honeyindexes $X_i = x_{i,1},x_{i,2},...x_{i,k}$. The login procedures are summarized below.

- User enters a password to login to the system.
- The administrator checks whether the hashed index of the entered password is available in the set of honeyindexes stored for the corresponding user. If not, the entered password is neither the correct password or the honeyword, login denied.
- Otherwise the system checks whether the hashed index is a sweetindex or a sugarindex.
- If it the sugarindex, then the system returns a TRUE value, otherwise returns a FALSE value and may also raise an alarm informing the administrator that the password file breach happened.

IV. SECURITY ISSUES

There are two major issues we need to consider. The first issue from [2] is the flatness of the honeyword generating algorithm. We suppose an adversary stolen a password file and if he could cracked the set of honeywords a particular user. Then he could easily find out the correct password among them, since all the honeywords except the correct password seems unreal. They are suggesting to refresh the passwords to prevent the brute-force attack. But account holders may not feel it as user friendly.

In case of Erguler method, the administrator of the system is using the passwords of other user to protect each users password. The users may feel that, their credentials are unsafe. Every user will be registering their accounts by accepting the terms and conditions of the applications. But the real fact is that most of the users registering their accounts by simply accepting the terms and conditions without even reading this. An adversary can use the sweetwords of a user on another systems which does not use honeywords. In any circumstances one should not use others passwords.

V. FLATNESS FROM UNREALISTIC HONEYWORDS

Our proposed model also based on honeywords to detect the password breaching. Instead of generating the honeywords from existing user passwords to achieve flatness, here we are making both the sweet word and sugarword unrealistic, so the attacker cannot distinguish between them. For achieving this, while registering the user is requested to add an image as his password.

Here we are including some new terms for the convenience of the discussion. Textual form of the secret image will be having two parts one is the sugarword which we taken as the user password and the rest can be called candyword. The server used to store the candyword can be called as honeystore.

The system first converts the image to text. Taking a string from it, hash it and store it on the server then the remaining part of the string what we called candyword is stored in another server (honeystore). Then k-1 set of honeywords are randomly generated, keep this along with the password string and store the third server and include the user name also on the same server. So when the adversary analyses the password files, she will recognize that each userid is paired with a set of honeywords. All seems unrealistic, even after decoding password; here we have created an uncertainty about the correct password. Here we have included two level of authentication checking while sign in and it is discussed in the coming sections.The contribution of our approach is that it is more secure and not invading the privacy of other users for the sake of a new user.

Not even a single honeyword is repeating in the system, so the adversary needs to perform a considerable amount of work for getting the password, even after this he will not be able to confirm the correct password since there are a number of honeywords attached to a single user. There can be situations where the adversary may be able to guess the correct password among the honeywords, but still the administrator has stored only a part of the textual form of the image in this server, the remaining part is safely hashed and stored in another server.

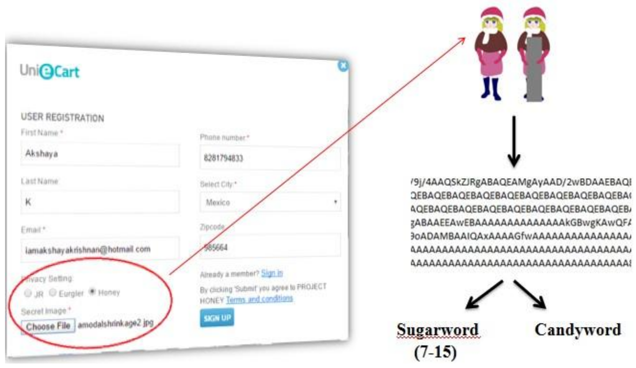
A. Initialisation

Here we need 3 servers for storing the credentials of users of the system as shown in Fig7. The first server is used to store the username and the corresponding honeyword set, ui,Xi, where ui is the username and the Xi is the hashed honeyword set, Xi = (x1,x2,x3,....xk). One among the list will be sugarword. It also contain a file with all the strings we have used to make the sugarwords for every users of the system.

The second server or the honeychecker will contain the indexes of all the users and the password strings. Then we need one more server to store the hashed values of the remaining portion of the text form of the image give as password by the user.

B. Registration

System is ready for registration after initialization process. Since we are using Legacy-UI for user to register the system. First get the password image from the user, then the system will convert it into an alphanumeric string. Taking initial contiguous set of characters from the string and the administrator of the system will treat this as the user's password, and the rest of the string (we call it candyword) is kept in a separate server(honeysore). The string taken as password is hashed and kept along with a set of randomly generated strings having length same as



UniCart

USER REGISTRATION

First Name *	Akshaya
Last Name	K
Email *	iamakshayakrishnan@hotmail.com
Phone number *	8281794833
Select City *	Mexico
Zipcode	985664
Privacy Setting	JR Eurgler [✓ Honey]
Secret Image *	Choose File amodalshrinkage2.jpg

Already a member? Sign in

By clicking 'Submit' you agree to PROJECT HONEY Terms and conditions

SIGN UP

/9j/4AAQSkZJRgABAQEAMgAyAAD/2wBDAAEBAQE
QEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQ
AQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEB
gABAAEEAwEBAAAAAAAAAAAAAAAkGBwgKAwQFA
BoADAMBAAIQAxAAAAGfwAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Figure 5. Registration schema of the proposed model.

the password, for each username. The correct password is kept in another list and the index of the correct password with its corresponding userid is kept in honeychecker.

Fig5 shows the registration form of the proposed model. Here the user can select in which way his passwords needs to be saved. There is an option called privacy setting with three radio buttons in it in the form. The first two radio buttons indicate the models developed by Juels-Rivest and Erguler (SectionIII) respectively. The third option indicate the proposed model. After selecting the option Honey, the user need to upload his secret image and submit the form to complete the registration. As soon as the user completes his registration, he will get a welcome mail from the system. The system will convert the secret image to textual form using Base64 algorithm. Then the system will select the sugarword from it and store the sugarword and the candyword in separate servers as shown in the Fig6 and Fig7.

C. Honeychecker

The auxiliary service honeychecker is used to store the secret information like the legitimate password id for each user account with their userid. The honeychecker communicate with the main server in an authenticated way. The two commands send to the honeychecker are

- Set : C_i,u_i where c_i is the index of the legitimate password for the user u_i
- Check : $u_{i,j}$
  Checks whether the password index c_i for the user user u_i is equal to given j.

That is, the auxiliary service honeychecker only knows what is the correct password index for the user, but not the password or the hashed password.

D. Login

A client can roll out improvements to his profile points of interest or do any approved procedure, simply after he is signed into the system. At the point when then sign in method begins, the system need to check whether the password is the legitimate password for the particular user account.

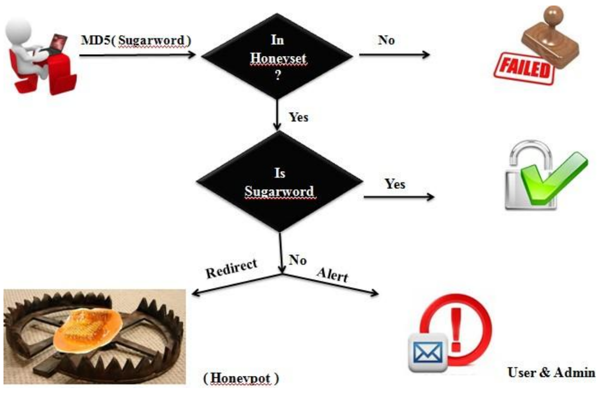


Figure 8. Login schema of the proposed model.

In our system the user will be asked to provide his secret image during sign in. The administrator will convert the secret image uploaded by the user in to its textual form and takes initial set of n characters from it and hash it using MD5 and check whether it is one among the honeywords for that user account. If not, it is neither the sugarword nor the sweetword, then the login fails. Otherwise the system will check whether it is a sweetword for that account. For this, index of the userid and password needs to be checked with the honeychecker. If the honeychecker has confirmed it is the sugarword then append that with its corresponding candyword stored in the honeystore for second level of confirmation. Then compared this whole string with the textual form of the image uploaded by the user while sign in to confirm it is the same. After this second level of confirmation the system will return the true result. Otherwise the user will be redirected to the honeypot account.

In the honeypot account the adversory won't have the capacity to play out any of the approved assignments and in the meantime the overseer will alarm the proprietor of the client record to change his secret image. The login schema is depicted in Fig8

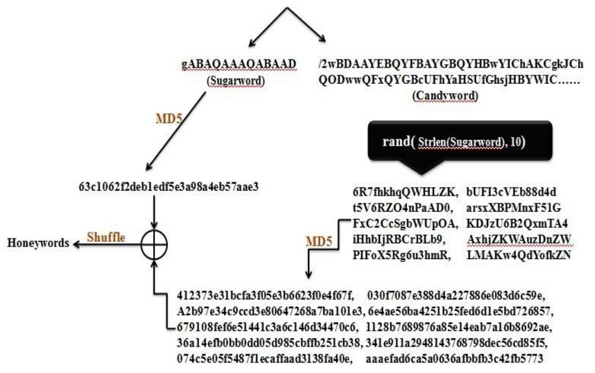
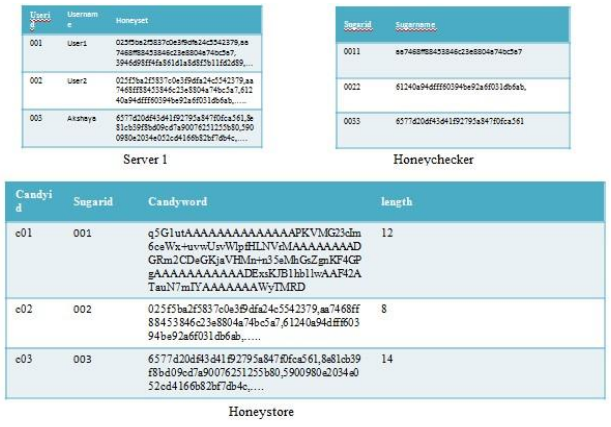

gABAQAAAQABAAD
(Sugarword)
/2wBDAAYEBQYFBAYGBQYHBwYIChAKCgkJCh QODwwQFxQYGBcUFhYaHSUfGhsjHBYWIC......
(Candyword)
MD5
rand(Strlen(Sugarword), 10)
63c1062f2deb1edf5e3a98a4eb57aae3
6R7fhkhqQWHLZK, bUFI3cVEb88d4d
t5V6RZO4nPaAD0, arsxXBPMnxF51G
FxC2CcSgbWUpOA, KDJzU6B2QxmTA4
iHhbIjRBCrBLb9, AxhjZKWAuzDnZW
PIFoX5Rg6u3hmR, LMAKw4QdYofkZN
Honeywords
Shuffle
MD5
412373e31bcfa3f05e3b6623f0e4f67f, 030f7087e388d4a227886e083d6c59e,
A2b97e34c9ccd3e80647268a7ba101e3, 6e4ae56ba4251b25fed6d1e5bd726857,
679108fef6e51441c3a6c146d34470c6, 1128b7689876a85e14eab7a16b8692ae,
36a14efb0bb0dd05d985cbffb251cb38, 341e911a2948143768798dec56cd85f5,
074c5e05f5487f1ecaffaad3138fa40e, aaaefad6ca5a0636afbbfb3c42fb5773

Figure 6. Suagarword and Sweetwords combines to form Honeywords.



Userid	Username	Honeyset
001	User1	025f5ba2f5837c0e3f9dfa24c5542379,aa7468ff88453846c23e8804a74bc5a7, 3946d98ff4fa861d1a8d8f5b11fd2d89,....
002	User2	025f5ba2f5837c0e3f9dfa24c5542379,aa7468ff88453846c23e8804a74bc5a7,61240a94dfff60394be92a6f031db6ab,.....
003	Akshaya	6577d20df43d41f92795a847f0fca561,8e81cb39f8bd09cd7a90076251255b80,5900980e2034e052cd4166b82bf7db4c,.....

Server 1

Sugarid	Sugarname
0011	aa7468ff88453846c23e8804a74bc5a7
0022	61240a94dfff60394be92a6f031db6ab,
0033	6577d20df43d41f92795a847f0fca561

Honeychecker

Candyid	Sugarid	Candyword	length
c01	001	q5G1utAAAAAAAAAAAAAAPKVMG23dm6ceWx+uvwUsvWlpfHLNVrMAAAAAAAADGRm2CDeGKjaVHMn+n35eMhGsZgnKF4GPgAAAAAAAAAAADExsKJB1hb11wAAF42ATauN7mIYAAAAAAAWyTMRD	12
c02	002	025f5ba2f5837c0e3f9dfa24c5542379,aa7468ff88453846c23e8804a74bc5a7,61240a94dfff60394be92a6f031db6ab,.....	8
c03	003	6577d20df43d41f92795a847f0fca561,8e81cb39f8bd09cd7a90076251255b80,5900980e2034e052cd4166b82bf7db4c,....	14

Honeystore

Figure 7. Storing the credentials in separate servers.

E. Change Password

When a user requested to change his password, the administrator needs to confirm whether he is the legitimate owner of that user account. The user needs to sign in to the system for changing his credentials. In case of changing the password the user will be asked to upload the current secret image also the new image. After confirming the current secret image and the userid with the honeychecker the user also needs to pass through phone number verification, in which the administrator will send a verification code via SMS to the phone number the user provided during registration. The user will be able to change the secret image only if he could submit the same verification code.

VI. SECURITY ANALYSIS OF THE PROPOSED MODEL

Here we are investigating the security of the proposed model.

One of the common problem related to password protection is the use of passwords co-related to username. This is not very easy to solve in usual password protection mechanisms. Since we are using image as password, this issue will not affect our system much.

- DoS Attack
  Denial of service is usually accomplished by flooding the targeted machine or resource with superfluous requests in an attempt to overload systems and prevent some or all legitimate requests from being completed. A DoS attack is similar to a group of people crowding the entry door to a shop, and not letting legitimate parties enter into the shop, disrupting normal operations.

- An adversary who is intended to perform DoS attack need not have the password file or its contents. Even if the adversary created some user accounts by himself, he cannot use the secret images of his user account to generate false alarm (as in case of Erguler model) as we are not generating honeywords from existing user passwords. If there are some situations in which the adversary guess any of the honeywords (rare chance) because we are using meaningless alphanumeric strings as honeywords), he cannot use it to generate false alarms, since none of the honeywords in the system repeats. as honeywords), he cannot use it to generate false alarms, since none of the honeywords in the system repeats.

- Password Guessing
  Here we assume that the adversary has attacked the main server and has plundered password files. He may be able to guess the sugarword of any of the accounts. But the word we treated as password is not the complete password. It need its candyword append to it to get the secret image.

- Brute-force Attack
  If adversary attempts to perform brute-fore attack, then he will hits the honeypot and the administrator detects about the password disclosure situation.

VI. SIGNIFICANCE OF THE PROPOSED MODEL

In this section, we are discussing about the significance our proposed model against the existing models with respect to DoS resistance, flatness, usability and storage.

A. DoS resistance

DoS attacks needs to be prevented for a secure password protection mechanism. Note that the authors[2] 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. However as stated in adaptation of a strong password composition policy likely prevents occurrence of common passwords in high numbers, i.e. probability of a common password is assigned as a honeyword for a specific user will be negligibly low. Although, an adversary may hit a real password using a common password in the system, it is not necessarily a honeyword for the corresponding account. The use of real passwords as honeywords will not be accepted by the users of the system. Thus converting the secret image of that user in to textual form and keeping one part of it with honeywords and the other part in another server will enhance the security of the system.

Last but not least issue is that in our proposed model in addition to honeywords, honeypots 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. In other words, in our approach administrator should take stronger actions in case of a honeypot attempt by entering with honeywords in order to diminish DoS vulnerability.

B. Flatness

For our proposed model as described previously a part of the textual form of the secret image and the meaningless random strings become honeywords for a user. Hence, our model satisfies perfect flatness as long as the secret image is not correlated with username. Investigation of a target user profile (age, gender, religion etc.) gives no advantage to an adversary in password guessing. Comparing our method with the other model, one can see that our method is better in terms of flatness: The honeywords it carry all characteristics of the real passwords in the same system.

C. Usability

In this part, we compare our approach with the earlier model in terms of practicality and ease of use. By considering the those model whose password list is constructed with composition of numerous real passwords and randomly generated passwords, one can argue about how the real password source is provided. If the same resource of real passwords is used in different sites, similar inherited weaknesses related to honeyword generation may be observed.

Nonetheless, if use of publicly available password lists is forbidden (as suggested by the authors), then it will not be easy to get required large number of real passwords. Conversely, our approach does not need to use an external real password resource in honeyword generation, rather it just feeds itself.

Therefore, we claim that our approach is simpler and more practical for implementation.

D. Storage

The traditional password system requires nearly hN storage for usernames, where N is the number of users in the system and h is the length of the password hash in bytes. The system in [2] requires khN storage, where k indicate the number of the sweetwords for each user account from the username will not change after adaptation of the honeywords. Since we are using MD5 the practical value of the h will be 16. Let us consider each index requires 4 bytes and if we neglected the storage cost of honeypots then the storage cost become :

$$4KN + hN + 4N + 4N \qquad (1)$$

Here the first 4*N* is for honeychecker and the second is for the honeystore. When comparing the storage to the method in [1], we give the ratio as :

$$\frac{4KN + hN + 4N + 4N}{4KN + hN + 4N} = \frac{4K + h + 8}{4K + h + 4} \quad (2)$$

This ratio is independent of the number of users and if the value of *k* and *h* are realistic it's value will be less than one.

E. Experimental Evaluation

We implement our honeymodel in Kohana framework, including separation of sugarword and candyword, sweetwords generation, MD5 hashing. We choose an Uniecommerce environment to set up our Honeymodel as its privacy is by means of passwords. The honeymodel is given as a privacy setting option in the registration form of the application. The next section will discuss about the performance evaluation of the Honeymodel.

1) Performance: Performance needs to be checked to determine how the system performs under certain workload in terms of responsiveness and stability. One of the important performance metric is average response time under normal and peek load, that is actual performance experienced by the users of the system. The response time a system acts upon the action of the users and the time taken to complete the specific amount of task also the load of the system such as the requests per second. The performance analysis of the honeymodel is depicted in Fig10.

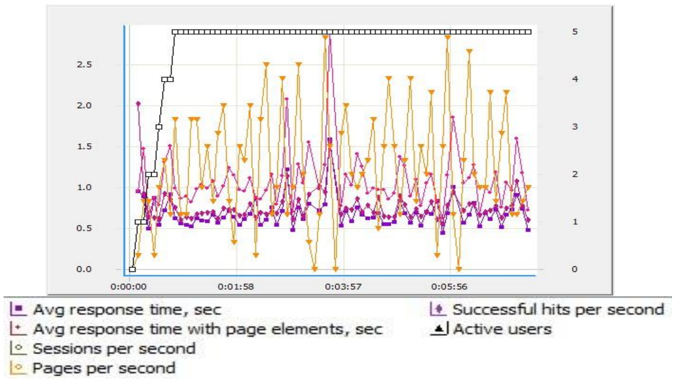


Figure 9. Performance of the Honeymodel.

Here we consider the average response time, average response time with page elements, sessions per second, pages per sec, successful hits per sec and a number of active users for analysing the performance. Response time is the time from the first byte of page request sent till the last received byte of server response. In other words, it is the time from clicking a link or a button in browser till the moment a page is downloaded. Response Time report represents the values of server response time. Average and maximum response times are the most important characteristics of load testing. They tell you how long a user waits for server response to his request. You should ensure that users of your site/application get the response in acceptable time. Response Time report represents the values of server response time. Average and maximum response times are the most important characteristics of load testing. They tell you how long a user waits for server response to his request. You should ensure that users of your site/application get the response in acceptable time.

There are 3 important limits for response time values:

- 0.1 second. It is an ideal response time. Users feel that the system is reacting instantaneously, and do not sense any interruption.
- 1.0 second. It is the highest acceptable response time. Users still do not feel an interruption, though they will notice the delay. Response times above 1 second interrupt user experience.
- 10 seconds. It is the limit after which response time becomes unacceptable. Moreover, recent studies show that if response time exceeds 8 seconds, user experience is interrupted very much and most users will leave the site or system.

Normally, response times should be as fast as possible. The interval of most comfortable response times is 0.1 - 1 second. Although people can adapt to slower response times, they are generally dissatisfied with the times longer than 2 seconds.

2) Bandwidth: Bandwidth of an application indicate the amount of data transfer between the user, application and internet also the level of traffic. Bandwidth is a very important entity for an on-line system. The bandwidth can also considered as speed of the system. Bandwidth has a close relation with the web traffic.

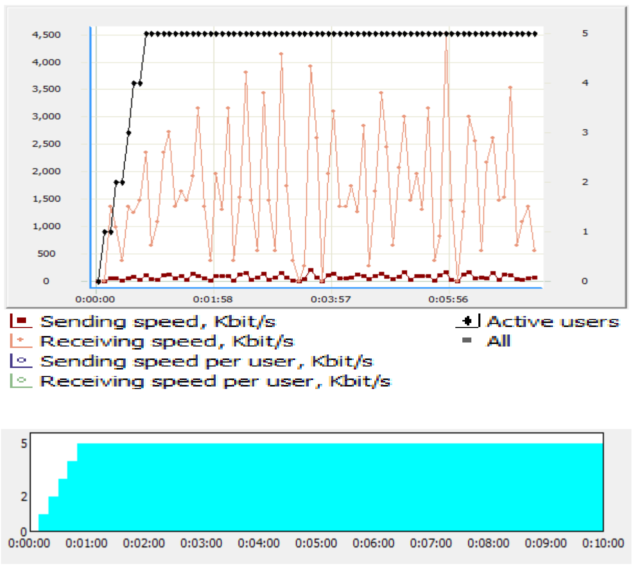


Figure 10. Bandwidth of the system with Honeymodel AND User load graph (X axis time, Y axis number of users).

More bandwidth may increase the traffic in the system. System bandwidth analysis is depicted in Fig9.

Bandwidth Usage report contains the information on server bandwidth: the speed of receiving the information from the server and the speed of sending the information to the server. This report will help you know whether the bandwidth of Internet connection to your server is sufficient to provide a required level of performance or not. Here we consider the following:

- KBytes sent : This shows how many KBytes were sent to the server at the specified periods of time.
- KBytes received : This shows how many KBytes were received from the server at the specified periods of time.
- Sending speed, kbit/s : This shows how many kbits per second were sent to the server.
- Receiving speed, kbit/s : This shows how many kbits per second were received from the server.
- Sending per user speed, kbit/s :This shows the sending speed per virtual user.
- Receiving per user speed, kbit/s : This shows the receiving speed per virtual user.

Streaming data, KBytes : This shows how many KBytes were received from the server at the specified periods of time for the streaming requests.

VIII. CONCLUSION

In this study we analysed that, honeyword mechanism not an ideal password protection mechanism or a password storage mechanisms, but this is an ideal mechanism for preventing the attackers from exploiting the stolen credentials. As the honeyword generation mechanism adapted will decide the strength of the honeyword system, we can say that our mechanism is a strong mechanism: because it achieves perfect flatness. Another point we would like to stress is that the low probability for DoS must be guaranteed by the system as it may become serious threat. Dos can be prevented by employing unpredictable honeywords.

We have compared the proposed model with other models with respect to DoS resistance, flatness, and usability has advantages over other models in terms of flatness, security and usability. Thus the model suggested by us is feasible, fair, legal and and reasonable.

IX. FUTURE WORK

In the future we would like to refine our model by including the option to make changes to the secret image at the time of registration to make it more secure. Taking the sugarword from random position of the textual form of the secret image for each user account can make the system more powerful.

ACKNOWLEDGMENT

The author would like to thank the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

[1] Imran Erguler, *Achieving Flatness*: *Selecting the Honeywords from Existing User Passwords,* IEEE Transactions on Dependable and Secure Computing 10.1109/TDSC.2015.2406707

[2] A. Juels and R. L. Rivest, *Honeywords*: *Making Password cracking Detectable,* in Proceedings of the 2013 ACM SIGSAC Conference on Computer Communications Security, ser. CCS 13. New York, NY, USA: ACM, 2013, pp. 145160. [Online].

[3] T. Wadhwa. *"Why your next phone will include fingerprint, facial, and voice recognition,"* Forbes, 29 March 2013.

[4] B.-A. Parnell. *"LinkedIn admits site hack, adds pinch of salt to passwords,"* The Register, 7 June 2012.

[5] I. Paul. *"Update: LinkedIn confirms account passwords hacked,"* PC World, 6 June 2012.

[6] Bojinov, E. Bursztein, X. Boyen, and D. Boneh, *Kamouflage: Lossresistant Password Management,* in Computer Security ESORICS 2010. Springer, 2010, pp. 286302.

[7] Shrisha Rao., *Data and system security with failwords.U.S. Patent Application,* US2006/0161786A1, U.S. Patent Office, July 20, 2006.

[8] Z. A. Genc, S. Kardas, and K. M. Sabir, *Examination of a New Defense mechanism: Honeywords,* International Journal of Engineering Trends and Technology , Volume-27 Number-4.

[9] M. Burnett, *The Pathetic Reality of Adobe Password Hints,* , 3rd ed. https://xato.net/windows-security/adobe-password-hints.

[10] E. Spafford. *"Observations on reusable password choices,"*In USENIX Security, 1992.