# Implementation of Selecting the Honeywords from Existing User Passwords

Ashish K.Bhise[1], Vidya Jagtap[2],

*[1] Ashish K.Bhise ME Student, Computer Department, G.H. Raisoni College of Engineering Chas Ahmednagar,Maharashtra, India*
*2 Vidya Jagtap Professor, Computer Department, G.H. Raisoni College of Engineering Chas Ahmednagar,Maharashtra, India*

## ABSTRACT

*Recently, proposed honeywords (decoy passwords) to detect attacks against hashed password databases. For every user account, the legitimate password is stored with several honeywords in order to sense impersonation. If honeywords are selected perfectly, a cyber-attacker who steals a file of hashed passwords can't be sure if it is a honeyword or the real password for any account. Moreover, entering with a honeyword to login will trigger an alarm inform the administrator about a password file break. At the expense of increasing the storage requirement by twenty times, the authors introduce effective and a simple solution to the detection of password file disclosure events. In this survey, we scrutinize the honeyword system and present some remarks to highlight possible weak points. Also, we suggest an alternative approach that selects the honeywords from existing user passwords in the system in order to supply realistic honeywords a perfectly flat honeyword generation method and also to reduce storage cost of the honeyword scheme.*

**Keyword:** - *Login, Authentication, honeywords, honeypot, passwords, password cracking.*

## 1. INTRODUCTION

Disclosure of password files is a severe security problem that has affected millions of companies and users like Yahoo, RockYou, eHarmony, LinkedIn, and Adobe [2], [1], since, leaked passwords make the users target of more possible cyber-attacks. These recent events have demonstrated that the weak password storage methods are currently in place on many web sites. For e.g. 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.]. Indeed, once a password file is stolen, by using the password cracking techniques like the algorithm of Weir et al. [4.] it is simple to capture most of the plaintext passwords.

In this respect, there are two problems that should be considered to overcome these security problems: First point, passwords must be protected by taking appropriate precautions and storing with hash values computed through salting or some other mechanisms. Hence for an adversary it must be difficult to invert hashes to acquire plaintext passwords. The second is that a secure system should detect whether a password file disclosure incident happened or not to take appropriate actions. In this survey, we aim on the latter issue and deal with fake passwords or accounts as a simple and cost effective solution to detect compromise of passwords. Honeypot is one of the methods to verify occurrence of a password database breach. In this approach, the administrator purposely creates fraud user accounts to lure adversaries and detects a password disclosure, if any one of the honeypot passwords get used [6], [5]. This concept has been modified by Florencio and Herley [7.] to safe online banking accounts from password brute-force attacks. According to the study, for every user incorrect login attempts with few passwords lead to honeypot accounts, i.e. malicious behaviour is recognized. Let's see an example, there are 108 possibilities for 8 digit password and let system links 10000 wrong password to honeypot accounts, so the adversary performing the brute-force attack 10000 times more likely to hit a honeypot account than the real account. Bojinov et al. in [8] was introduced Use of decoys for building theft-resistant called as Kamouflage. In this system model, the fake password sets are stored with the real user password set to conceal the actual passwords, thereby forcing an adversary to carry out a considerable amount of online work before getting the correct information. Recently, Rivest and Juels have presented the honeyword mechanism to detect an adversary who attempts to login with cracked passwords [9.]. Basically, for every username a set of sweetwords is constructed such that only one element is the correct password and the others are honeywords. Hence, when an adversary tries to enter into the system with a honeyword, an alarm is triggered to inform the administrator about a password leakage. The description of the method will be given in the next section.

In this study, we analyze the honeyword approach and give some remarks about the security of the system. Furthermore, we notice that the key item for this method is the generation algorithm of the honeywords such that they shall be indistinguishable from the right 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, this technique also decreases the storage cost compared with the honeyword method in [9.].

## 2. LITERATURE SURVEY

### 2.1 Examination of a new defence Mechanism: Honeywords

It has become much easier to crack a password hash with the advancements in the graphical processing unit (GPU) technology. An adversary can hack a user's password using brute-force attack on password hash. Once the password has been recovered no server can detect any illegitimate user authentication (if there is no extra mechanism used). In this context, recently, Rivest and Juels published a paper for improving the security of hashed passwords. Roughly saying, they propose an approach for user authentication, in which some fake passwords, i.e., "honeywords" are added into a password file, in order to detect impersonation. Their solution includes an auxiliary secure server called "honeychecker" which can distinguish a user's actual password among her honeywords and immediately sets off an alarm whenever a honeyword is used. In this paper, we examine the security of the proposal, provide some possible improvements which are very easy to implement and introduce an enhanced model as a solution to an open problem.

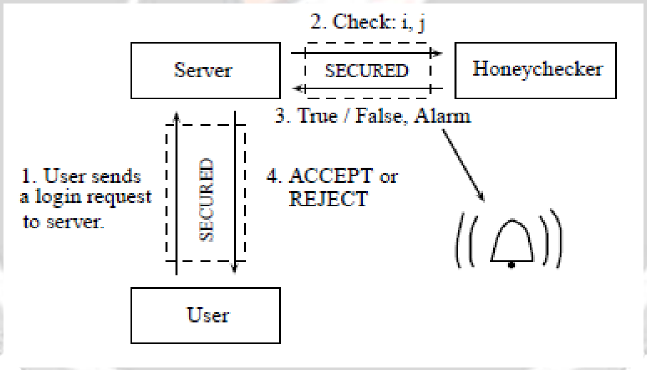

**Fig -1:** Login schema of a system using honeywords.

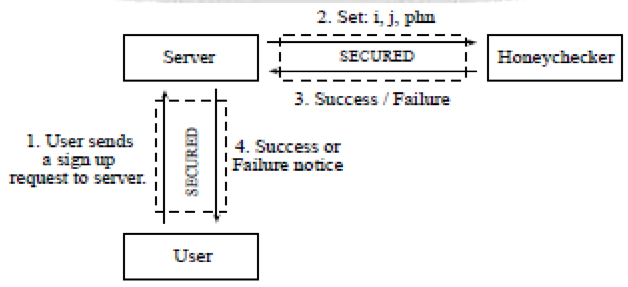

**Fig -2:** Sign up schema of an enhanced honeywords system.

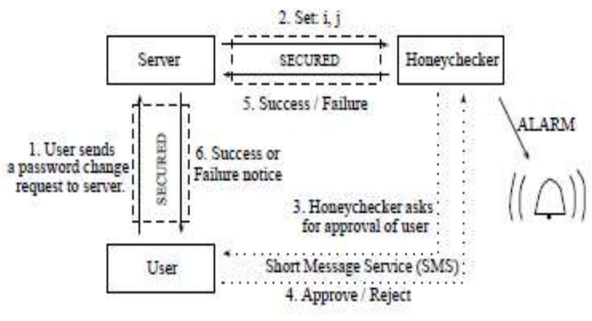
2. Set: i, j
Server
SECURED
Honeychecker
5. Success / Failure
ALARM
1. User sends a password change request to server.
SECURED
6. Success or Failure notice
3. Honeychecker asks for approval of user
User
Short Message Service (SMS)
4. Approve / Reject

**Fig -3:** Password change schema of an enhanced honeywords system.

### 2.2 Improving Security using Deception.

As the convergence between our physical and digital worlds continues at a rapid pace, much of our information is becoming available online. In this section we develop a novel taxonomy of methods and techniques that can be used to protect digital information. We discuss how information has been protected and show how we can structure our methods to get better results. We explore complex relationships among protection techniques ranging from denial and isolation, to degradation and obfuscation, through deception and negative information, ending with adversary attribution and counter-operations. Here we show analysis of these relationships and describe how they can be applied at different scales within organizations. We also verify some of the areas that are worth further investigation. These protection techniques can be mapped against the cyber kill-chain model and discuss some findings.

Moreover, we analyze the use of deceptive information as a useful protection method that can significantly enhance the security of systems. We examine advantages these techniques can have when protecting our information in addition to traditional methods of hardening and hiding. In deceptive information, we not only lead attackers astray, but also give organizations the ability to find leakage; create doubt and uncertainty in any leaked data; add risk at the adversary's side to using the leaked information; and significantly improve our abilities to attribute adversaries. We examine how to overcome some of the challenges that hinder the adoption of deception-based techniques and present some recent work, our own contribution, and some promising directions for future research.

### 2.3 The Dangers of Weak Hashes

There have been several high publicity password leaks over the past year including LinkedIn, eHarmony and Yahoo. Where ever you never want to have vulnerabilities that allow hackers to get access the password hashes, you also want to make sure that if the hashes are compromised it is not very easy for hackers to generate passwords from the hashes and these leaks have demonstrated, large companies are using weak hashing mechanisms that make it easy to crack user passwords.

### 2.4 Explicit Authentication Response Considered Harmful.

Proposed system, password guessing attacks shows that most users authentication techniques provide a yes/no answer as authentication process and these attacks are somewhat restricted by Automated Turing Tests (ATTs, example Captcha challenges) that attempt to mandate human assistance. ATTs are not very difficult for legalize users, but always pose an inconvenience. Many ATT implementations are also found to be vulnerable to improved image processing algorithms. ATTs can be made more difficult for automated software, but that is limited by the trade-off between user-friendliness and effectiveness of ATTs. As attackers gain control of large-scale botnets, relay the challenge to legalize users at compromised websites, or even have ready access to cheap, sweat-shop human solvers for defeating ATTs. Using deception techniques (as in honeypots), we aim the user-verifiable authentication scheme (Uvauth) that tolerates, instead of detecting or counteracting, guessing attacks. Uvauth provides access to all authentication attempts; the right password enables access to a legitimate session with valid user data, and all incorrect passwords lead to fake sessions. Authorized users are expected to learn the authentication outcome implicitly from the present user data, and are relieved from answering ATTs the authentication result never leaves the server and thus remains (directly) inaccessible to attackers. In addition, we advise using adapted distorted images and pre-registered images/text as a complement to convey an authentication response, especially for accounts that do not host much personal data.

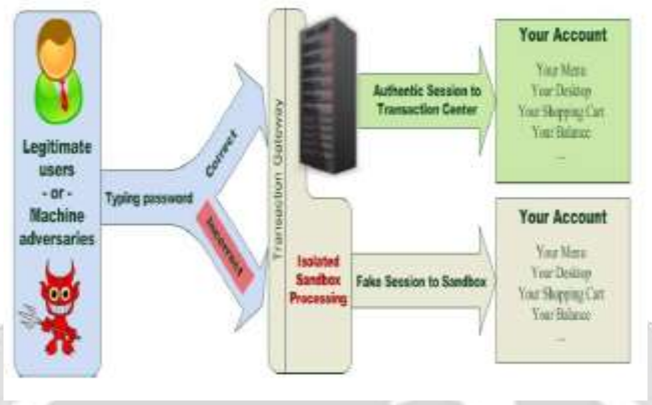

**Fig -3:** Overview of user-verifiable authentication

### 2.5 Investigating the Distribution of Password Choices

The distributed passwords chosen by users have implications for site security, password-handling algorithms and even how users are permitted to select passwords. There are lots of passwords list and using these password lists from four various websites; we investigate if Zipf's law is a good description of the frequency with which passwords are chosen. We use various standard statistics, which measure security of password distributions, if modeling the data using a simple distribution is effective. We then consider how much the password distributions from every site have in common, using password cracking as a metric. This shows that these distributions have enough high-frequency passwords in common to provide effective speed-ups for breaking passwords and finally, as an alternative to a deterministic banned list.

## 3. PROPOSED SYSTEM

In this system we presented the honeyword mechanism to detect an adversary who attempts to login with cracked passwords.

Firstly, users enter his/her credentials to the system. After entering password system generate hash values of the password with some additional fake hash values, which are already exist in the system database of various users password.

For every username a set of sweetwords is constructed such that only one element is the correct password and the others are honeywords (decoy passwords).

After entering honeywords we add index values in the passwords, which we call honeyindexes.

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.

Hence, when an adversary tries to enter into the system with a honeyword, alarm is triggered to notify the administrator about a password leakage.

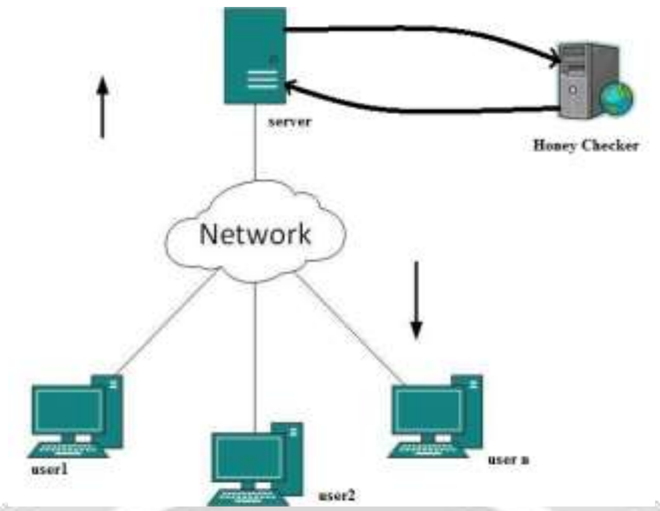

**Fig -5:** Proposed system architecture

The proposed system modules and there details are as shown in the given below:

**Admin Module:**
This module works as controller and authoriser module for the users being connected to the Server. As the server is a controlling module, it can view the current user status by analysing their behaviour. The Admin module can also block the users is it is malicious or seems to attack other users.

**User Module:**
The user module is the one that logs in to the system uses server services. The user does not play a major role in the proposed system as the proposed system is server operation centric and hence focuses over the server operations.

**Server Module:**
The Server Module is the one where the hash or Honeyword generation algorithm works and creates various honeywords from the existing realistic passwords. The server checks the index of the input password hash and the matching hash from the list, and sends it to the honeychecker module for authenticating the index of the input hash value.

**Honeychecker Module:**
The honeychecker module is basic module but a very vital module which stores the actual index ci of the correct hash value and computes the comparison of the index value received from the server and returns the Boolean value. If the value matches, it returns TRUE else it returns FALSE.

**Algorithm**

**A. Encryption Algorithm**

**Encryption:**

**Step 1:** Let the message to be transmitted be ”CRYPTOGRAPHY”.
First find the ASCII equivalent of the above characters.
C R Y P T O G R A P H Y
67 82 89 80 84 79 71 82 65 80 72 89

**Step 2:** Add these numbers with the digits of the Whole number as follows
67 82 89 80 84 79 71 82 65 80 72 89
(+) 1 5 3 1 25 9 1 125 27 1 5 3
68 87 92 81 109 88 72 207 92 81 77 92

**Step 3:** Convert the above data into a matrix as follows:

$$A=\begin{bmatrix} 68 & 81 & 72 & 81 \\ 87 & 109 & 207 & 77 \\ 92 & 88 & 92 & 92 \end{bmatrix}$$

**Step 4:** Consider an encoding matrix...

$$B=\begin{bmatrix} 1 & 5 & 3 \\ 1 & 25 & 9 \\ 1 & 125 & 27 \end{bmatrix}$$

Step 5: After multiplying the two matrices (B X A) we get The encrypted data is

$$C=\begin{bmatrix} 779 & 890 & 1383 & 742 \\ 3071 & 3598 & 6075 & 2834 \\ 13427 & 16082 & 28432 & 12190 \end{bmatrix}$$

779, 3071, 13427, 890, 3598, 16082, 1383, 6075, 28431742, 2834, 12190
The above values represent the encrypted form of the given message or original information.

**Decryption:**
Decryption involves the process of getting back the original data using decryption key.

**Step 1:** Original data decryption, Perform the encoding matrix inversion. D=

$$(-1/240)*\begin{bmatrix} -450 & 240 & -30 \\ -18 & 24 & -6 \\ 100 & -120 & 20 \end{bmatrix}$$

**Step 2:** Multiply the decoding matrix with the encrypted data. DXC=

**Step 3:** Transform the above result as given below:
68 87 92 81 109 88 72 207 92 81 77 92

$$\begin{bmatrix} 68 & 81 & 72 & 81 \\ 87 & 109 & 207 & 77 \\ 92 & 88 & 92 & 92 \end{bmatrix}$$

**Step 4:** Subtract with the digits of the Whole numbers as follows
68 87 92 81 109 88 72 207 92 81 77 92

(-) 1 5 3 1 25 9 1 125 27 1 5 3

---

67 82 89 80 84 79 71 82 65 80 72 89

The ASCII conversion to character again gives the actual decrypted data i.e.

C R Y P T O G R A P H Y in this example.

## 4. MATHEMATICAL MODEL

The proposed system can be mathematically formulated as follows:

Firstly, T fake user accounts (honeypots) are created with their passwords (see Appendix A for details).

Also an index value between [1;N], but not used previously is 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 = (xi;1; xi;2; : : : ; xi;k); one of the elements in Xi is the correct index (sugarindex) as ci.

Now, we use two password files as F1 and F2 in the main server: F1 stores username and honeyindex set, < hui;Xi > pairs as shown in Table 2, where hui denotes a honeypot account. Note that each entry has two elements. The first one is the username of the account and the second element is honeyindex set for the respective account.

Also, the table is sorted alphabetically by the username field. On the other hand, F2 keeps the index number and the corresponding hash of the password, < ci;H(pi) >, as depicted in Table 3. In this case, each entry in the table has two elements.

The first element is the sugarindex of the account and the second one is the hash of the corresponding password. Notice that the table is sorted according to the index values. Let SI denote the index column and SH represent the corresponding password hash column of F2.

Then the function f(ci) that gives password hash value in SH for the index value ci can be defined as: f(ci) = fH(pi) 2 SH :< ci;H(pi) > stored pair of ui and ci 2 SIg.

For each user ui, the sweetword list Wi is generated using the honeyword generation algorithm Gen(k). This procedure takes input k as the number of sweetwords and outputs both the password list Wi = (wi;1;wi;2; : : : ;wi;k) and ci, where ci is the index of the correct password (sugarword). The username and the hashes of the sweetwords as < ui; (vi; 1; vi; 2; ::: ; vi; k) > tuple is kept in the database of the main server, whereas ci is stored in another server called as honeychecker. By diversifying the secret information in the system - storing password hashes in one server and ci in the honeychecker - makes it harder to compromise the system as a whole, i.e. providing a basic form of distributed security [9]. Notice that in a traditional password technique < ui;H(pi) >pair is stored for each account, while for this system < ui; V i > tuple is kept in the database, where Vi = (vi;1; vi;2; : : : ; vi;k). The login procedure of the scheme is summarized below:

User ui enters a password g to login to the system.

Server firstly checks whether or not H(g) is in list Vi.
If not, then login is denied.

Otherwise system checks to verify if it is a honeyword or the correct password.

Let v(i;j) = H(g). Then j value is delivered to the honeychecker in an authenticated secure communication.
The honeychecker checks whether j = ci or not.

If the equality holds, it returns a TRUE value, otherwise it responses FALSE and may raise an alarm depending on security policy of the system.

## 5. RESULTS

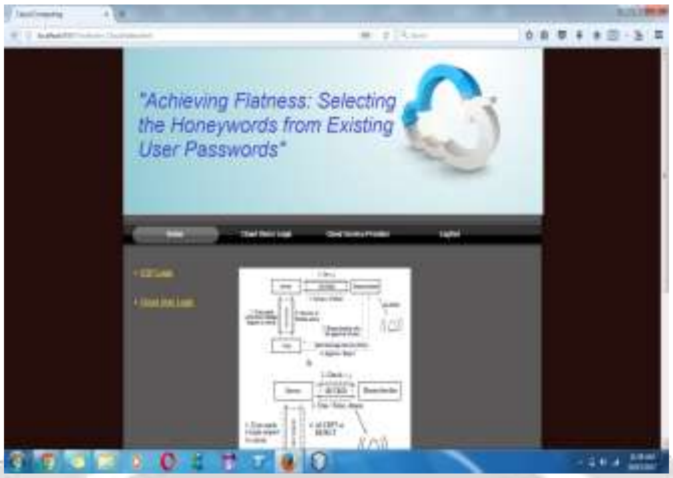
"Achieving Flatness: Selecting the Honeywords from Existing User Passwords"

**Fig -6:** Home Page

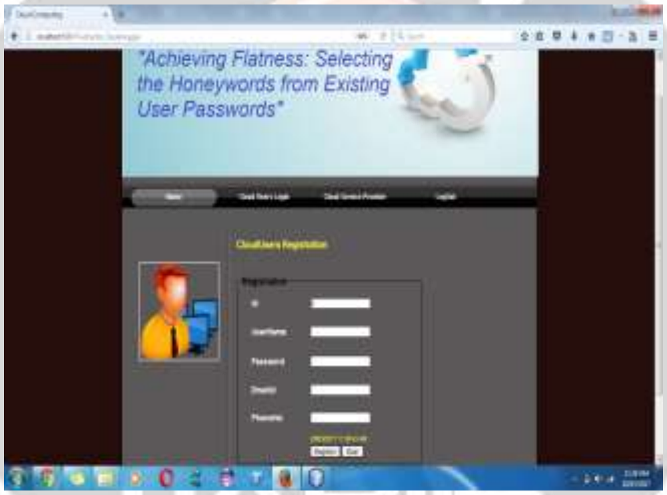
"Achieving Flatness: Selecting the Honeywords from Existing User Passwords"
CloudUsers Registration
Registration

**Fig -7:** Registration Page

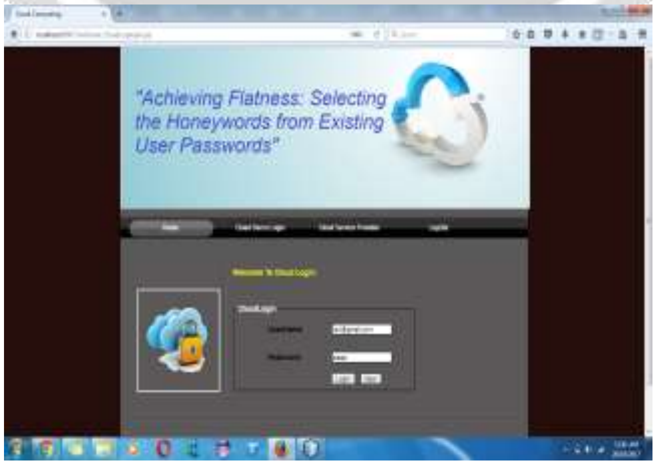
"Achieving Flatness: Selecting the Honeywords from Existing User Passwords"

**Fig -8:** Login Page

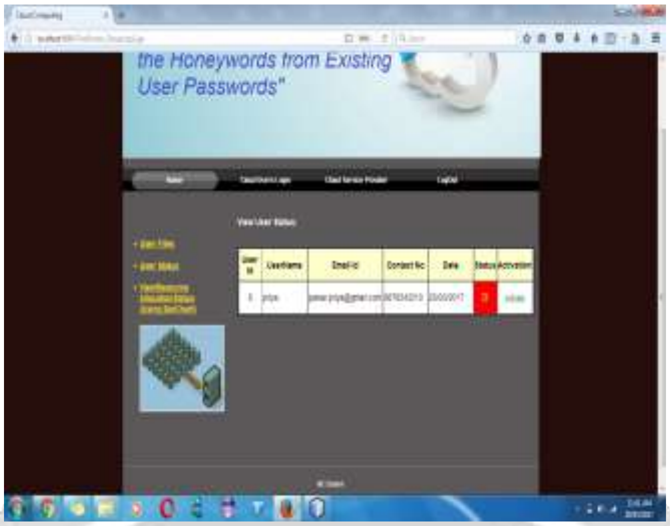

**Fig -9:** CSP User Activation Page

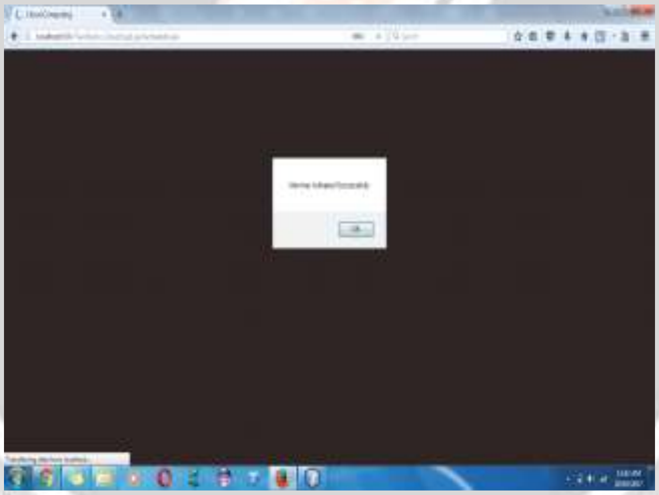

**Fig -10:** User Activated Successfully

## 6. RESULT ANALYSIS

| Result Description | Expected Output | Actual Output |
|---|---|---|
| The proposed system is based on authentication based system. In which user have to login with their valid user name and password. If someone who are not a valid user and he/she trying to login then the system not allowed the user to login with the system. | Suppose an invalid user is trying to login with the hash value with user-name and password then the system should raise the alarm. | The actual output of the proposed system is that when an invalid user or hacker logged the system with the hash value of valid user then the sys-tem raised the alarm to the admin. |

**Fig -10:** Result Analysis

## 7. CONCLUSION

In this study, we have analyzed the security of the honeyword system and addressed a number of flaws that need to be handled before successful realization of the scheme. In this respect, we have pointed out that 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 right password out of respective sweetwords. Another point that we would like to stress is that defined reaction policies in case of a honeyword 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. To combat such a problem, also known as DoS resistance, low probability of such an event must be guaranteed. This can be achieved by employing unpredictable honeywords or altering system policy to minimize this risk. Hence, we have noted that the security policy should strike a balance between DoS vulnerability and effectiveness of honeywords. Furthermore, we have demonstrated the weak and strong points of each method introduced in the original study. It has been shown that DoS resistance of the chaffing-by-tweaking method is weak and also its flatness can be questioned by regarding Remark 1. Although some weaknesses of the chaffing-by-tweaking techniques are accepted by their creators, we believe that it should not be considered as alternative method due to its predictable nature and a potential DoS weakness. Moreover, the chaffing-withtough nuts model has been investigated, and we have doubted about its favour as opposed to ideas of Juels and Rivest. On the other hand, the chaffing-with-a-password model can fulfill its claims provided that the generator algorithm is flat. Nevertheless, how the source of the real passwords is attained for this model should be answered before judging its applicability. Finally, we have presented a new approach to make the generation algorithm as close as to human nature by generating honeywords with randomly picking passwords that belong to other users in the system. We have compared the proposed model with other methods with respect to DoS resistance, flatness, 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, flatness and usability.

## 8. CONCLUSION

[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]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

[11]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

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