International Journal of Future Generation Communication and Networking
Vol. 13, No.2s, (2020), pp. 1720–1726

Honeyword Security using EBCDIC

Sumit Pravin Rathi[#1], Patel Rutu Manish[#2], Arivoli A.[*3]

School of Computer Science and Engineering, Vellore Institute of Technology, Vellore

Abstract

Password breach is a frequent and common phenomenon in software applications. These breaches are sometimes undiscovered for years. Many times, the users as well as companies are unaware of these unhandled breaches and they are not much interested in reporting or building up against them. Hence, there is a requirement for a robust system that could detect these breaches efficiently. Honeyword generation is one such technique to mitigate the risk of password breaches. Honeywords are fake, hashed character strings that are stored in databases which makes it difficult for the attacker to predict the correct password. As it is essential for a system to withstand brute force attack and provide better security, we propose honeyword generation approach using EBCDIC values which will randomly generate numbers to form sweet words, which is a file with all the honeywords and valid passwords altogether.

Keywords*— honeyword, hashing, password prevention, brute force, targeted password guessing*

I. INTRODUCTION

Juels and Rivest (in ACM-CCS 2013) suggested a password breach detection technique called honeywords. The basic idea is to generate multiple fake passwords based on the original password and store all of them together. These false passwords as well as valid password may be stored in the database or a file. The intruder may be able to reach out to the honeyword file successfully, but when reverse hash is performed on all the passwords on the list, it is difficult to guess the correct password. If brute force attack is implemented along all the honeywords then, the system will generate an alert to the respective user and admin. Also, legitimate users are not known to the honeywords, so any login attempt using the honeywords is identified as compromise to the database system. This way the risk of password compromise is minimized. In the proposed approach, each character of password is converted to its EBCDIC value and the same is done for the generated honeywords. EBCDIC stands for Extended Binary Coded Decimal Interchange Code; it is an 8-bit character encoding that comprises of alphanumeric characters. It is an enhancement of Binary Coded Decimal number system. EBCDIC is developed and used by IBM. The 8 bits are divided into two parts as zone and digit, each of 4 bits. Zone is used to identify the category, while digit bits identify the exact character. The prime responsibilities of the honey checker algorithm here are honeyword generation using EBCDIC conversion, and alerting users upon brute force attempts.

II. BACKGROUND STUDY

A comprehensive study in 2013 found that password breaches is a security problem that should be taken care of with proper effective measures. Organizations like Apple, LinkedIn, eHarmony, Adobe and many more have been victims to password breach in the past. Data and details of thousands of users was compromised in spite of having strong hashing techniques [12]. Akshima, Donghoon Chang et al. (2019) have compared the existing algorithms for honeyword generation which includes Tweaking, password-model, tough nuts, take-a-tail, close number formation, Caps-key based approach, modified-tail approach and Paired Distance Protocol. The algorithms are compared on the basis of certain factors which includes flatness, DoS resistant, Typo safety, Legacy UI, MSIO resistance, MSII resistance and user friendliness. The authors have proposed new models for honeyword generation, namely evolving-password model, the user-profile model and the append-secret model. The attack models proposed by the authors are "Multiple System Intersection attack considering Input" and "Multiple System Intersection attack considering Output". The evolving-password model focuses on flatness. It manages the probability of how much the honeywords are distinguished from original password. The user-profile model manages the honeyword generation by following token based approach. Token arrays are created and honeywords are generated using

ISSN: 2233-7857 IJFGCN

different combinations of tokens from each array. Also, the distance between honeyword and valid password is calculated which is ideally set to 3, and the distance should be greater than 3. The Append-secret model is a function that takes the password, a parameter l as length and r as a random character string as inputs. The function outputs a honeyword which is stored in the database along with valid password. The above proposed models are tested against various attacks like Targeted Password Guessing, Brute Force, Dos, MSII and MSIO and experimental results show how it overcomes them [1]. The proposed methodology of honeyword generation by Komal Naik et al. (2017) focuses on ways to overcome the inversion attacks. As the passwords and hash are stored in the database, it is easy to crack through the system and apply inversion model to get the original password in decoded form. To prevent this kind of attack an algorithm is proposed which does tweaking of characters. The original password is considered and certain characters are tweaked, the resultant characters are then mapped to numbers in a particular range. This formatted string is assigned ASCII values to generate the honeyword. The research also includes a survey of different methods to generate honeywords and their limitations [2].

As the storage requirement rises with increasing number of honeywords for each user, it is important to manage this issue and find alternatives which can save a considerable amount of space. In 2016, Imran Erguler came up with a solution to this limitation of generating greater number of honeywords. He has proposed an algorithm that uses character strings from the user details of all the users that are stored in the same database. This way, one can create more realistic honeywords which will make the guessing tough. This way the flatness parameter increases considerably and proves a better honeyword generation technique [3]. The research work of Khin Su Myat Moe and Thanda Win (2017) focuses on improved hashing techniques that can withstand brute force attack on passwords stored in database. Moreover, it addresses problems of large storage requirements, typo safety and old password management policies. A unique hashing scheme is used to store passwords in the database. The time complexity of hashing and salting algorithm is very less as compared to the Advanced Encryption Standard (AES) and Data Encryption Standard (DES).The comparative study for flatness, typo safety and storage overhead shows that the proposed algorithm is better than Chaffing by tweaking, Paired distance protocol, take a tail and password model. The storage overhead is only (1*n) while other algorithms have ((k-1)*n) storage overhead, where k is number of sweet words in a password file and n is number of users. However, the time complexity of the hashing algorithm increases with increase in length of original password [4]. In 2017, Nilesh Chakraborty and Samrat Mondal contributed to the honeyword generation research to address storage overhead, password prevention and certain usability issues. They proposed a protocol called Paired Distance Protocol which stores honeywords virtually. Here, public key encryption is used to overcome all the security pitfalls. As the authors have compared, this algorithm provides perfect flatness. Also, the user can login to different accounts with same credentials, thus it provides a user-friendly scheme. However, the user needs to remember extra two bits (i.e. password tail) along with username and password for login activity. This can be considered as a small disadvantage of the proposed technique [5]. Neelam C. More et al. (2016) proposed a Honey Encryption technique that maps the index to valid passwords. The procedure of this ideology is divided into four parts. Firstly, two files are created, one of which stores the honeyword ID and its index while the other has the honeyword index and the hashed value of valid password. An index generator is used to create the index. When a login attempt is made, the username-password are matched with the index and in turn, index is matched with the data in second file, the user will be notified if the attempt was unsuccessful. As stated, the honeywords can be generated using Chaffing by tweaking or chaffing by password model or chaffing by tough nuts or hybrid methodology [6]. Vruhali Thakur, Akash Kumble, et al. (2019) have proposed a honeyword generation system that does not allow unauthorized access to the system. When the hacker tries to login with any of the honeyword, he/she is redirected to a fake web page, which he/she may think as a legitimate page and successful login. Simultaneously the user is informed about this login attempt. In this way, it protects the user details as well as makes it difficult for the hacker to identify id the login is actually successful or not [7].

ISSN: 2233-7857 IJFGCN

III. PROPOSED METHODOLOGY

A. Honeyword Generation algorithm:

Nowadays, internet is one of the daily needs for the major growth in society. Internet is used by every individual nowadays. The security of every individual's data is very important and to protect the data is biggest challenge we are facing. To access specific website, we need to login with unique Username and Password. The Username and Password gets store in the database with various other details such as First Name, Last Name, Email ID, Phone Number, Date of Birth, etc. We need Username and Password to sign in and to access the resources such as email, social network, business sites, etc. In this honeyword generation method we are proposing four different ways to generate honeywords, calculate its hash and store them in the database as shown in Fig. 1.

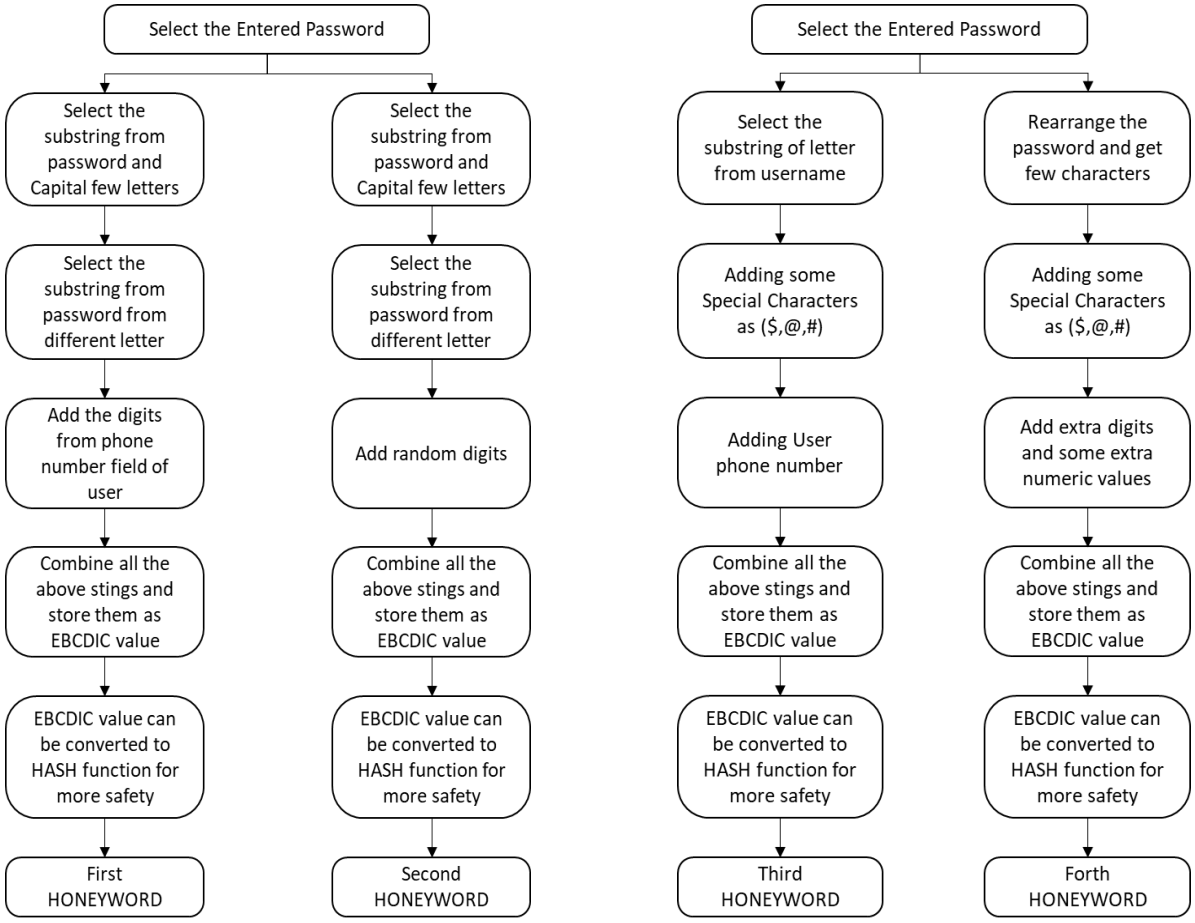

Fig. 1 Procedure for generation of four honeywords

Below are some conversion values of characters, numbers and some special characters

TABLE I
CONVERSION OF EBCDIC VALUES

EBCDIC Value	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4
ASCII Value	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D
Character	A	B	C	D	E	F	G	H	I	J	K	L	M

EBCDIC Value	D5	D6	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9

ISSN: 2233-7857 IJFGCN

ASCII Value	4E	4F	50	51	52	53	54	55	56	57	58	59	5A
Character	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

EBCDIC Value	F0	F2	F3	F4	F5	F6	F7	F8	F9	5A	5B	6D	7C
ASCII Value	30	32	33	34	35	36	37	38	39	5D	24	5F	40
Character	0	2	3	4	5	6	7	8	9	!	$	_	@

Here is an example of working of the above stated methodology.

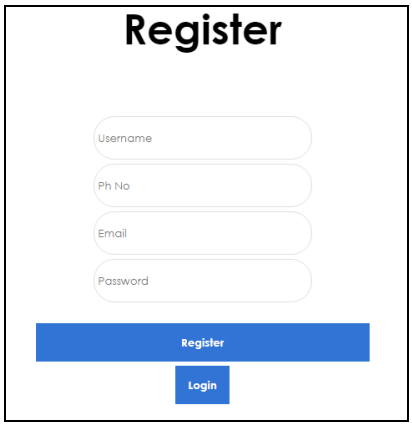

Fig. 2 Registration Form

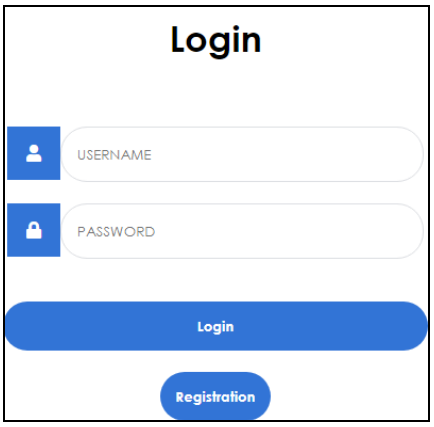

Fig. 3 Login Form

TABLE II
REGISTRATION FORM USER DETAILS

Username	abcde_vwxyz1289
Phone Number	9988776655
Email ID	abc_xyz1289@testmail.com
Password	HelloWorld@2020#

As discussed above, all characters of original password are converted to EBCDIC Hex. Same process is executed for each and every honeyword.

TABLE III
CONVERSION TO EBCDIC VALUES

H	**e**	**l**	**l**	**o**	**W**	**o**	**r**	**l**	**d**	**@**	**2**	**0**	**2**	**0**	**#**
C7	85	93	93	96	E6	96	99	93	84	7C	F2	F0	F2	F0	7B

As shown in Table IV, the honeywords are generated with various substring from password and username with adding digits, some special characters and extra alphabets are inserted randomly in the honeyword as shown in fig. 1. All the four passwords are converted into EBCDIC characters Hexadecimal value. Further they are hashed to store in the database.

TABLE IV
EBCDIC VALUES AS STORED IN DATABASE

Original Password	HelloWorld@2020#

ISSN: 2233-7857 IJFGCN

EBCDIC Value	C785939396E6969993847CF2F0F2F07B
Honeyword 1	LLello99887766
Honeyword 2	LOl978
Honeyword 3	de_$&~9988776655
Honeyword 4	lloW1998

While login, the password entered by the user will be compared with the password from the database. The password string from database will be decoded from hash and converted from EBCDIC value to ASCII value for comparison with user entered password. If the user entered password and original stored database password matches then user will be able to login and access the resources as shown in Fig. 4.

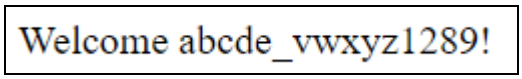

Fig. 4 Output of successful login

If the username is incorrect or the password mismatches then the incorrect username or incorrect password popup will be triggered. Consider a situation where the Hacker managed to get access to the user credential database, then hacker will try to decode and get the login details for portal. In this method, for security purpose we have generated four honeyword password strings. If the hacker tries to login with any of the fake password (i.e. honeyword) then he will be able to login with error message as "Access Denied" and email will be triggered to the user and admin of the database regarding such misbehaviour with credential database as shown in Fig. 5. This will help to manage and secure the credentials database.

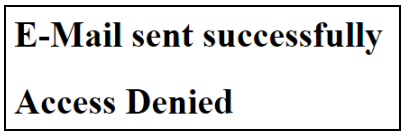

Fig. 5 Output when honeyword is used

IV. SECURITY ANALYSIS

A. BRUTE-FORCE ATTACK:

For this attack, assume that honey checker is not compromised and the intruder gets access to credential database from server. The probability of success for attacker to fetch or to obtain real password increases on the basis of flatness of the honeyword generated. In this scenario, the 'user-profile' model is used where honeyword is generated from some data of user-details, this technique will provide approximately perfect flatness for password to be unique or to be different from original password as the probability of credential related to user details is very high. Therefore, here Brute-Force attack do not help to uniquely identify the password.

B. TARGETED PASSWORD GUESSING:

In this attack, user details of user may help an intruder to distinguish an actual password from honeyword. In case of evolving-password model, if the password is related to user details then it is easy to identify the original password from honeywords. Password guessing is more effective on weak passwords. We have identified the issue and implemented the password with user-data model and we have rearranged the string of password. For additional security purpose we have added external randomly generating numbers and some special characters to change the entire string of password from getting identified. This will help for strong as well as weak passwords to be identified easily by intruder.

C. DENIAL-OF-SERVICE ATTACK

ISSN: 2233-7857 IJFGCN

Honeyword generation for 'user-profile' model requires user personal information. It is not difficult to fetch user personal data for attacker; data can be fetched using various social engineering attacks or with some Machine Learning Algorithms. This makes attackers task easy to fetch password from honeyword and carry out DoS attack on the system. To keep safe from such attacks we have rearranged the strings in the implementation as well as we have added some extra substrings to the password. The implementation of honeywords also added few more numeric values externally with some Special characters which helps to change the password and to protect the system from DoS attack.

V. CONCLUSIONS

In this work, we implemented a new way to generate honeyword to overcome several limitations of the existing system. For security purpose we have entirely manipulated the password and stored it in the database. Before storing password in database, it gets converted from characters to EBCDIC value set. The converted EBCDIC string is hashed with hash function and then stored in database. Our method produces honeywords that are indistinguishable from the password and achieve 'approximate flatness'. This secures the system from intruder to access the user data or user profile on web-world. Further we can add more honeywords to database as well as we apply techniques such as Levenshtein distance, Chaffing by tweaking, Close-number-formation (CNF) and various other techniques for generation of honeywords.

ACKNOWLEDGMENT

We would like to show our gratitude towards our primary supervisor Prof. Arivoli A. from School of Computer Science and Engineering at Vellore Institute of Technology, Vellore for offering us deep insight into the study and guiding us through the implementation of our idea.

REFERENCES

[1] Chang, Donghoon, Aarushi Goel, Sweta Mishra, and Somitra Kumar Sanadhya. "Generation of secure and reliable honeywords, preventing false detection." IEEE Transactions on Dependable and Secure Computing 16, no. 5 (2018): 757-769.

[2] KomalNaik, Ms, and Vinayak D. Shinde. "Honeyword Generation Approach Using ASCII Values for Security Enhancement." In International Conference On Emanations in Modern Technology and Engineering (ICEMTE-2017), vol. 5, no. 3, pp. 110-113.

[3] Erguler, Imran. "Achieving flatness: Selecting the honeywords from existing user passwords." IEEE Transactions on Dependable and Secure Computing 13, no. 2 (2015): 284-295.

[4] Moe, Khin Su Myat, and Thanda Win. "Improved hashing and honey-based stronger password prevention against brute force attack." In 2017 International Symposium on Electronics and Smart Devices (ISESD), pp. 1-5. IEEE, 2017.

[5] Chakraborty, Nilesh, and Samrat Mondal. "On designing a modified-UI based honeyword generation approach for overcoming the existing limitations." Computers & Security 66 (2017): 155-168.

[6] Neelam .C. More, Minaj .M. Pathan, Mahesh .B. Totre and Asst. Prof. Swati. S. Gore. "Honeyword:Achiving secure Passwords using HoneyEncryption." International Journal of Advance Engineering and Research Development Volume 3, Issue 11, November -2016

[7] Prof. Vruhali Thakur, Akash Kamble, Abhishek Hatankar, Rushikesh Jagdale and Ankita Talekar "Honeywords: The New Approach for Password." Security." International Journal for Research in Applied Science & Engineering Technology (IJRASET) ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887 Volume 7 Issue IV, Apr 2019

[8] Kumar, Sanjeev, B. Janet, and R. Eswari. "Multi-Platform Honeypot for Generation of Cyber Threat Intelligence." In 2019 IEEE 9th International Conference on Advanced Computing (IACC), pp. 25-29. IEEE, 2019.

[9] Matin, Iik Muhamad Malik, and Budi Rahardjo. "Malware Detection Using Honeypot and Machine Learning." In 2019 7th International Conference on Cyber and IT Service Management (CITSM), vol. 7, pp. 1-4. IEEE, 2019.

ISSN: 2233-7857 IJFGCN

[10] Nursetyo, Arif, Eko Hari Rachmawanto, and Christy Atika Sari. "Website and Network Security Techniques against Brute Force Attacks using Honeypot." In 2019 Fourth International Conference on Informatics and Computing (ICIC), pp. 1-6. IEEE, 2019.
[11] Ahn, Seonggwan, Thummin Lee, and Keecheon Kim. "A Study on Improving Security of ICS through Honeypot and ARP Spoofing." In 2019 International Conference on Information and Communication Technology Convergence (ICTC), pp. 964-967. IEEE, 2019.
[12] Mirante, D. and Cappos, J., 2013. Understanding password database compromises. Dept. of Computer Science and Engineering Polytechnic Inst. of NYU, Tech. Rep. TR-CSE-2013-02.

ISSN: 2233-7857 IJFGCN