JES
Journal of Engineering Sciences

Vol 13, Issue 05, May /2022

ISSN NO:0377-9254

Authentication by Encrypted Negative Password

S.Indu, K.Radha Krishna
DEPARTMENT OF MCA
Sri Padmavathi College Of Computer Sciences & Technology

ABSTRACT:

Secure password storage is a vital aspect in systems based on password authentication, which is still the most widely used authentication technique, despite its some security flaws. In this paper, we propose a password authentication framework that is designed for secure password storage and could be easily integrated into existing authentication systems. In our framework, first, the received plain password from a client is hashed through a cryptographic hash function (e.g., SHA-256). Then, the hashed password is converted into a negative password. Finally, the negative password is encrypted into an Encrypted Negative Password (abbreviated as ENP) using a symmetric-key algorithm (e.g., AES), and multi-iteration encryption could be employed to further improve security. The cryptographic hash function and symmetric encryption make it difficult to crack passwords from ENPs. Moreover, there are lots of corresponding ENPs for a given plain password, which makes precomputation attacks (e.g., lookup table attack and rainbow table attack) infeasible. The algorithm complexity analyses and comparisons show that the ENP could resist lookup table attack and provide stronger password protection under dictionary attack. It is worth mentioning that the ENP does not introduce extra elements (e.g., salt); besides this, the ENP could still resist precomputation attacks. Most importantly, the ENP is the first password protection scheme that combines the cryptographic hash function, the negative password and the symmetric-key algorithm, without the need for additional information except the plain password.

I. INTRODUCTION

O WING to the development of the Internet, a vast number of online services have emerged, in which password authentication is the most widely used authentication technique, for it is available at a low cost and easy to deploy. Hence, password security always attracts great interest from academia and industry. Despite great research achievements on password security, passwords are still cracked since users' careless behaviors. For instance, many users often select weak passwords they tend to reuse same passwords in different systems. they usually set their passwords using familiar vocabulary for its convenience to remember. In addition, system problems may cause password compromises. It is very difficult to obtain passwords from high security systems. On the one hand, stealing authentication data tables (containing usernames and passwords) in high security systems is difficult. On the other hand, when carrying out an online guessing attack, there is usually a limit to the number of login attempts. However, passwords may be leaked from weak systems. Vulnerabilities are constantly being discovered, and not all systems could be timely patched to resist an attack, which gives adversaries an opportunity to illegally access weak systems. In fact, some old systems are more vulnerable due to their lack of maintenance. Finally, since passwords are often reused, adversaries may log into high security systems through cracked passwords from systems of low security. After obtaining authentication data tables from weak systems, adversaries can carry out offline attacks. Passwords in the authentication data table are usually in the form of hashed passwords. However, because processor resources and storage resources are becoming more and more abundant, hashed passwords cannot resist precomputation attacks, such as rainbow table attack and lookup table attack. Note that there is a trend of generalization of adversaries, because anyone could obtain access to information on vulnerabilities from vulnerability databases, such as the Open Source Vulnerability Database (OSVDB), National Vulnerability Database (NVD), and the Common Vulnerabilities and

Exposures (CVE) [19], and then make use of these information to crack systems. Moreover, they could download and use attack tools without the need for very professional security knowledge. Some powerful attack tools, such as hashcat [20], RainbowCrack [21] and John the Ripper [22], provide a variety of functions, such as multiple hash algorithms, multiple attack models, multiple operating systems, and multiple platforms, which raises a higher demand for secure password storage. In these situations, attacks are usually carried out as follows. First, adversaries precompute a lookup table, where the keys are the hash values of elements in a password list containing frequently-used passwords, and the records are the corresponding plain passwords in the password list. Next, they obtain an authentication data table from low security systems. Then, they search for the plain passwords in the lookup table by matching hashed passwords in the authentication data table and the keys in the lookup table. Finally, the adversaries log into higher security systems through cracked usernames and passwords, so that they could steal more sensitive information of users and obtain some other benefits. A considerable number of attacks are carried out in this way, so that adversaries could obtain passwords at a low cost, which is advantageous to their goals. One of the main reasons for the success of the above lookup table attack is that the corresponding hashed password is determined for a given plain password. Therefore, the lookup table could be quickly constructed, and the size of the lookup table could be sufficiently large, which results in a high success rate of cracking hashed passwords. Typical password protection schemes include hashed pass-word, salted password and key stretching. Among these schemes, hashed password would be gradually eliminated for its vulnerability for precomputation attacks. Although salted password could resist precomputation attacks, it introduces an extra element (i.e., salt) and could not resist dictionary attack. In addition, salt tends to be implemented by mistake (such as salt reuse and short salt). Key stretching schemes, such as bcrypt [23], scrypt [24] and Argon2 [25] (the winner of Password Hashing Competition [26]), are used to defend against dictionary attack. Although key stretching schemes provide stronger password protection than salted password under dictionary attack, they impose an extra burden on programmers for configuring more parameters. In addition, they also use salt to resist precomputation attacks. Besides these schemes, some other password protection schemes were proposed. In [17], a scheme based on MD5 was proposed. It is a variant of salted password, where the salt is two random strings. Although it could resist lookup table attack and make dictionary attack difficult, it introduces many parameters, which makes it complicated and inconvenient to use. In [27], dynamic salt generation and placement are used to improve password security. Essentially, this scheme is also a variant of salted password, where the salt is a random string that is dependent on the original password. Consequently, it could resist lookup table attack, however it could not defend against dictionary attack and also introduces an extra element (i.e., salt). In [28], improved dynamic Key-Hashed Message Authentication Code function (abbreviated as d-HMAC) was proposed for password storage. It is also a variant of salted password, where the salt is the user's public key, and it introduces a secret key, which makes it inconvenient to use. In summary, although some new password protection schemes were proposed, they are similar to typical password protection schemes essentially. Therefore, in Section VI, without loss of generality, we only compare the typical password schemes with our scheme. In this paper, a password protection scheme called Encrypted Negative Password (abbreviated as ENP) is proposed, which is based on the Negative Database (abbreviated as NDB) [29]–[32], cryptographic hash function and symmetric encryption, and a password authentication framework based on the ENP is presented. The NDB is a new security technique that is inspired by biological immune systems [29] and has a wide range of applications [33]–[36]. Symmetric encryption is usually deemed inappropriate for password protection. Because the secret key is usually shared by all encrypted passwords and stored together with the authentication data table, once the authentication data table is stolen, the shared key may

be stolen at the same time [37]. Thus, these passwords are immediately compromised. However, in the ENP, the secret key is the hash value of the password of each user, so it is almost always different and does not need to be specially generated and stored. Consequently, the ENP enables symmetric encryption to be used for password protection. As an implementation of key stretching [38], multi-iteration encryption is introduced to further improve the strength of ENPs. Compared with the salted password scheme and key stretching, the ENP guarantees the diversity of passwords by itself without introducing extra elements (e.g., salt).

II. SYSTEM ANALYSIS

EXISTING SYSTEM:

The existing system actually uses the simplest mechanism of all the other techniques. The plain password is just encrypted and stored in the database. This mechanism is highly insecure and you can also find that it is easy to attack and get the password. The other main mechanism which is used till date is the hashing mechanism where in the plain password is hashed using hashing algorithms such as the Secure Hash Algorithm or the Message Digest Algorithm. Comparing to the previous mechanism it provides more security and also it doesn't provide the actual password but the hashed value of the password. But the plain password can be from the hashed value from the rainbow table attack and lookup table attack. Thus to reduce the vulnerability and risk we are using the Encrypted Negative Password System for the file Management system.

DISADVANTAGES OF EXISTING SYSTEM:

- Hashed Password: The simplest scheme to store passwords is to directly store plain passwords. However, this scheme presents a problem that once adversaries obtain the authentication data table, all passwords are immediately compromised.
- Salted Password: To resist precomputation attacks, the most common scheme is salted password [17]. In this scheme, the concatenation of a plain password and a random data (called salt) is hashed through a cryptographic hash function.
- Key Stretching: To resist dictionary attack, key stretching [38], which converts weak passwords to enhanced passwords, was proposed. Key stretching could increase the time cost required to every password attempt, so that the power of defending against dictionary attack is increased.

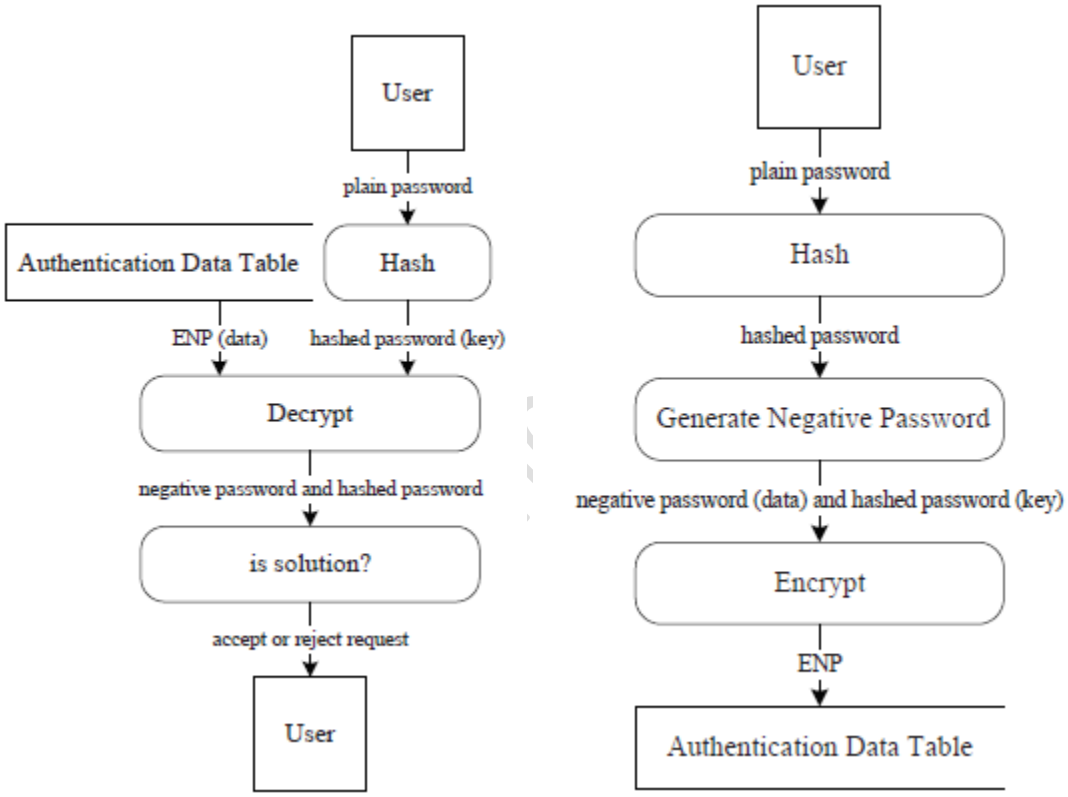
PROPOSED SYSTEM:

The proposed framework includes two phases: the registration phase and authentication phase. When adopting our framework to protect passwords in an authentication data table, the system designer must first select a cryptographic hash function and a symmetric-key algorithm, where the condition that must be satisfied is that the size of the hash value of the selected cryptographic hash function is equal to the key size of the selected symmetric-key algorithm. For convenience, some matches of cryptographic hash functions and symmetric-key algorithms are given in Table II. In addition, cryptographic hash functions and symmetric-key algorithms that are not listed here could also be used in the ENP, which adequately indicates the flexibility of our framework. The proposed framework is based on the ENP.

ADVANTAGES OF PROPOSED SYSTEM:

- We propose a password protection scheme called ENP, and we propose two implementations of the ENP: ENPI and ENPII, including their generation algorithms and verification algorithms. Furthermore, a password authentication framework based on the ENP is presented.
- We analyze and compare the attack complexity of hashed password, salted password, key stretching and the ENP. The results show that the ENP could resist lookup table attack without the need for extra elements and provide stronger password protection under dictionary attack.

III. SYSTEM ARCHITECTURE:



IV. IMPLEMENTATION

MODULES:-

- Registration Phase
- Authentication Phase
- Encrypted Negative Password (ENP)
- NDB generation algorithm

MODULES DESCRIPTON:-

Registration Phase

- ✓ On the client side, a user enters his/her username and password. Then, the username and plain password are transmitted to the server through a secure channel;
- ✓ If the received username exists in the authentication data table, "The username already exists!" is returned, which means that the server has rejected the registration request, and the registration phase is terminated; otherwise, go to Step (3);
- ✓ The received password is hashed using the selected cryptographic hash function;
- ✓ The hashed password is converted into a negative password using an NDB generation algorithm (i.e., Algorithm A.1 or Algorithm A.2 in the Appendix);
- ✓ The negative password is encrypted to an ENP using the selected symmetric-key algorithm, where the key is the hash value of the plain password. Here, as an additional option, multi-iteration encryption could be used to further enhance passwords;
- ✓ The username and the resulting ENP are stored in the authentication data table and "Registration success" is returned, which means that the server has accepted the registration request.

Authentication Phase

- ✓ On the client side, a user enters his/her username and password. Then, the username and plain password are transmitted to the server through a secure channel;
- ✓ If the received username does not exist in the authentication data table, then "Incorrect username or password!" is returned, which means that the server has rejected the authentication request, and the authentication phase is terminated; otherwise, go to Step (3);
- ✓ Search the authentication data table for the ENP corresponding to the received username;
- ✓ The ENP is decrypted (one or more times according to the encryption setting in the registration phase) using the selected symmetric-key algorithm, where the key is the hash value of the plain password; thus, the negative password is obtained;
- ✓ If the hash value of the received password is not the solution of the negative password (verified by Algorithm 1 or Algorithm 2), then "Incorrect username or password!" is returned, which means that the server has rejected the authentication request, and the authentication phase is terminated; otherwise, "Authentication success" is returned, which means that the server has accepted the authentication request.

ENCRYPTED NEGATIVE PASSWORD

- ✓ ENPs could be obtained plain password (i.e., a sequence of characters) from a client is first hashed using a cryptographic hash function.
- ✓ Next, the hashed password is converted into a negative password using an NDB generation algorithm (i.e., Algorithm A.1 or Algorithm A.2 in the Appendix).
- ✓ Then, the negative password is encrypted using a symmetric-key algorithm. Thus, the ENP is obtained. The solution of the negative password is the hash value of the received plain password.

NDB Generation algorithm

- ✓ The NDB generation algorithm is selected for converting a hashed password to the corresponding negative password.

- ✓ The NDB generation algorithm is a one-to-many mapping; simultaneously, it is reversible; additionally, while keeping the one-to-many relationship, it does not introduce extra elements (such as salt). Specifically, given a hashed password, there are lots of corresponding negative passwords; a negative password has one and only one corresponding hashed password; this conversion is done by the NDB generation algorithm itself, and not dependent on extra elements.

V. CONCLUSION

In this paper, we proposed a password protection scheme called ENP, and presented a password authentication framework based on the ENP. In our framework, the entries in the authentication data table are ENPs. In the end, we analyzed and compared the attack complexity of hashed password, salted password, key stretching and the ENP. The results show that the ENP could resist lookup table attack and provide stronger password protection under dictionary attack. It is worth mentioning that the ENP does not need extra elements (e.g., salt) while resisting lookup table attack.

REFERENCES

[1] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, "Passwords and the evolution of imperfect authentication," Communications of the ACM, vol. 58, no. 7, pp. 78–87, Jun. 2015.
[2] M. A. S. Gokhale and V. S. Waghmare, "The shoulder surfing resistant graphical password authentication technique," Procedia Computer Science, vol. 79, pp. 490–498, 2016.
[3] J. Ma, W. Yang, M. Luo, and N. Li, "A study of probabilistic password models," in Proceedings of 2014 IEEE Symposium on Security and Privacy, May 2014, pp. 689–704.
[4] A. Adams and M. A. Sasse, "Users are not the enemy," Communications of the ACM, vol. 42, no. 12, pp. 40–46, Dec. 1999.
[5] E. H. Spafford, "Opus: Preventing weak password choices," Computers & Security, vol. 11, no. 3, pp. 273–278, 1992.
[6] Y. Li, H. Wang, and K. Sun, "Personal information in passwords and its security implications," IEEE Transactions on Information Forensics and Security, vol. 12, no. 10, pp. 2320–2333, Oct. 2017.
[7] D. Florencio and C. Herley, "A large-scale study of web password habits," in Proceedings of the 16th International Conference on World Wide Web. ACM, 2007, pp. 657–666.
[8] R. Shay, S. Komanduri, A. L. Durity, P. S. Huh, M. L. Mazurek, S. M. Segreti, B. Ur, L. Bauer, N. Christin, and L. F. Cranor, "Designing password policies for strength and usability," ACM Transactions on Information and System Security, vol. 18, no. 4, pp. 13:1–13:34, May 2016.
[9] D. Wang, D. He, H. Cheng, and P. Wang, "fuzzyPSM: A new password strength meter using fuzzy probabilistic context-free grammars," in Proceedings of 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Jun. 2016, pp. 595–606.
[10] H. M. Sun, Y. H. Chen, and Y. H. Lin, "oPass: A user authentication protocol resistant to password stealing and password reuse attacks," IEEE Transactions on Information Forensics and Security, vol. 7, no. 2, pp. 651–663, Apr. 2012.
[11] M. Zviran and W. J. Haga, "Password security: An empirical study," Journal of Management Information Systems, vol. 15, no. 4, pp. 161– 185, 1999.
[12] P. Andriotis, T. Tryfonas, and G. Oikonomou, "Complexity metrics and user strength perceptions of the pattern-lock graphical authentication method," in Proceedings of Human Aspects of Information Security, Privacy, and Trust. Springer International Publishing, 2014, pp. 115– 126.