International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 04 Issue: 12 | Dec-2017 www.irjet.net p-ISSN: 2395-0072

A Survey of Password Attacks and Safe Hashing Algorithms

Tejaswini Bhorkar

Student, Dept. of Computer Science and Engineering, RGCER, Nagpur, Maharashtra, India

---

Abstract - *To authenticate users, most web services use pair of username and password. When a user wish to login to a web application of the service, he/she sends their user name and password to the web server, which checks if the given username already exist in the database and that the password is identical to the password set by that user. This last step of verifying the password of the user can be performed in different ways. The user password may be directly stored in the database or the hash value of the password may be stored. This survey will include the study of password hashing. It will also include the need of password hashing, algorithms used for password hashing along with the attacks possible on the same.*

Key Words: Password, Hashing, Algorithms, bcrypt, scrypt, attacks, SHA-1

1.INTRODUCTION

Web servers store the username and password of its users to authenticate its users. If the web servers store the passwords of its users directly in the database, then in such case, password verification is just comparison of two strings. However, this is extremely risky and irresponsive approach as the attacker who gains access to the server database has access to the passwords of all the users. Thus, now-a-days websites stores the hash value of the password of its users instead of storing the password directly. The goal of password hashing is to prevent an attacker to read the passwords even if he/she gains access to the database.

1.1 Hashing

Hashing is a type of algorithm that takes data of any size and converts it into data of fixed size. The main difference between hashing and encryption is that a hash is irreversible. Hash functions are used for hashing. A hash function is any function that can be used to map data of arbitrary size to data of fixed size. The output of the hash function is called hash codes, hash values, hash sums, or hashes.

A hash function should satisfy the following:

a) Two different messages should not have the same hash value. Thus, the hash function should be resistant against collision.

b) Given a hash value, it should be difficult or practically impossible to generate the corresponding message. Thus, the hash function should have pre-image resistance.

c) Given a message, it should be infeasible to find another message that would generate the same hash as that of the first message. Thus, the hash function should be resistant to second-pre-image.

1.2 Hashing Algorithms

Different hash algorithms are used in order to generate hash values. Some commonly used hash functions include the following:

- MD5
- SHA-1
- SHA-2
- SHA-3

MD5:

This algorithm takes data of arbitrary length and produces message digest of 128 bits (i.e. 16 bytes). In this algorithm, the input message is broken into chunks of 512-bit blocks. The message is padded by a 1 followed by zeros so that the message length is 64 bits less than a multiple of 512. The rest of the bits are filled with 64 bits representing to the length of original message. This hashing algorithm is broadly utilized but it is inclined to collisions. However, in practise, the collision attack is too slow to be useful. This is broken in regard to collisions but not in regard to pre-images or second-pre-images.

SHA-1:

Secured hash function 1 is a hash algorithm that takes a string of any length and reduces it to a message digest of 160 bits. In this algorithm, message is "padded" by a 1 and followed by as many 0's required to make the message length equal to 64 bits less than an even multiple of 512. 64-bits indicating the length of original message are appended to the end of padded message. Padded message is processed in 512-bit blocks. Cryptographic shortcomings have been found for SHA-1, and along these lines, the standard was never again endorsed for most cryptographic users after 2010.

SHA-2:

Secured hash function 2 is a hash algorithm that takes a string of any length and reduces it to a message digest. The SHA-2 family comprises of six hash functions with hash values that are

224, 256, 384 or 512 bits: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, and SHA-512/256. In this algorithm, message is "padded" with a 1 and as many 0's as necessary to bring the message length to 64 bits less than an even multiple of 512. 64-bits indicating the length of original message are appended to the end of padded message. Padded message is processed in 512-bit blocks.

SHA-3:

SHA-3 standard was released on 5th of august 2015. SHA3 uses sponge construction which has two phases, absorbing phase and squeezing phase. The message blocks must be XORed into subset in the absorbing phase, which is then transformed. In the squeezed phase, output blocks are read from the same subset of state, alternated with state transformations. Different variants of SHA-3 include SHA3-224, SHA3-256, SHA3-384, and SHA3-512 producing message digest of 224,256,384 and 512 bits and uses block size of 1152, 1088, 832, and 576 bits respectively.

2. Password Attacks

Most web servers force their users to generate password which is at least 8 characters long and contain letters, numbers, signs, and at least one capital letter. This is because, in order to prevent attackers to identify user's password, we should force the users to select a strong password apart from choosing a good hashing function. There are different strategies to attack user passwords. Some of them include the following:

a. Dictionary attacks: With a dictionary attack, the attacker will try to use word list, these can consist of mostly used words, passwords, years, names, etc. For each word, the attacker will run the hash algorithm and see if the generated hash is same as the hash in the database. If this is the case, then the attacker knows the word from which the hash was derived is the password.

b. Brute force: With brute force attack, the attacker will try all the possible combinations of characters. When using password of at least 8 characters long, only using the ASCII character set, there are 128^8 possibilities of passwords.

c. Lookup Tables: The general idea is to pre-compute the hashes of the passwords in a password dictionary and store them, and their corresponding password, in a lookup table data structure. Lookup tables are great way for cracking many hashes of a similar sort quickly. A good implementation of a lookup table can process hundreds of hash lookups per second, even when they contain many billions of hashes.

Rainbow tables: Rainbow tables are a time-memory trade-off technique. They are similar to lookup tables, except that the hash cracking speed is compromised to reduce the size of lookup tables. Because they are smaller, the solutions to more hashes can be stored in the same amount of space, making them more effective. Rainbow tables are capable of cracking any md5 hash of a password which is up to 8 characters long exist.

2.2 Need for Hashing & its Properties

If an attacker gets a red access to our database, we don't want him to retrieve the passwords plaintext hence we hash passwords. We often store usernames, email addresses and other personal information in our database. Security rule #1 dictates that users need to be protected from themselves. We can make them aware of the risks, we can tell them not to re-use passwords, but we all know that in the end there will still be people who use the same password for their Facebook, Gmail, LinkedIn and corporate email. What you do not want is that when the attacker gets his hand on your database, he immediately has access to all the above accounts (usernames/email addresses will be the same). Hashing passwords is to keep this from happening, when the attacker gets his hands on your database; you want to make it as difficult as possible to redeem those passwords using a brute-force attack. Hashing passwords won't make your site any more secure, however it will perform harm regulation in case of a breach.

Properties-

Following are the properties that should be fulfilled by password hashing:

a) In order to prevent brute force attack of compromising the data in one run, password hashing should have a unique salt per password.

Salt and Pepper: Before hashing a password, a unique value, known as salt, is appended (depending on the algorithm) to the password. Salt is used to prevent attack using rainbow tables. Information provided by the user should not be used in the generation of salt as this will result in the generation of same hash value from different databases given the same user information and password.

Pepper is a secret value or key that is used to turn the hash into a HMAC. HMAC is a hash function. However, it cannot be reproduced without knowing the key. Thus, this increases the security if the attacker has access to database but doesn't have the access to the place where the key is stored. If the key can be obtained by the attacker, then, the pepper will not add any security to the hash function.

b) Executing the function once must be faster i.e. the algorithm should be fast on software.

c) In order to prevent brute force attack on distributed systems, execution of the function concurrently should be slow i.e. the function should be slow on hardware.

Thus, from the above properties, it is clear that a password hashing algorithm should be slow in order to prevent brute force attack. This eliminates the use of MD5, SHA-1, SHA-2 and SHA-3 hash algorithms. Below algorithms can be used for password hashing.

2.3 Safe Password Hashing Algorithms

Following are the three algorithms that are safe to use for password hashing:

a) PBKDF2: This algorithm is used to derive keys. Due to the property of being slow, the PBKDF2 algorithm is used for hashing passwords even though it was not intended for password hashing. The resulting derived key can be used to securely store password. The following is done by PBKDF2 in order to derive a key:

K = PBKDF2 (PRF, Password, Salt, c, dkLen)

where,
DK: Derived key.
PRF: Preferred HMAC function.
c: amount of iterations.
dkLen: length of delivered key.

A salt should be at least 64-bits in length and the minimum amount of iterations should be 1024. The algorithm applies HMAC function and repeats the calculations 1024 times of the result. This means that hashing the password is 1024 times slower. However, when brute forcing on distributed systems, this algorithm does not offer a lot of protection.

b) bcrypt: Currently the default secure standard for password hashing is bcrypt. It is derived from the Blowfish block cipher. Blowfish block cipher uses loop up tables, which are initially in memory, to generate the hash. Bcrypt has been around for 14 years, based on a cypher that is that has been around for over 20 years. This algorithm has been well tested and hence is considered as the standard of password hashing.

c) scrypt: scrypt is a hashing algorithm which has same properties as bcrypt, except for that when the number rounds increase, the computation time and memory space required to produce hash increments exponentially. Scrypt concentrates on operations that are hard for anything else than a computer, for example, random memory access. Scrypt was created in response to the increasing attack on bcrypt. Scrypt uses snapshots that are storage of series of intermediate state data. These snapshots are used in further derivation operations. These snapshots, that are stored in memory, grow exponentially when a round is increased. So, when a round is added, it makes the brute force attack on the password exponentially harder. Scrypt has been around for only couple of years and thus is still relatively new compared to bcrypt.

3. CONCLUSIONS

This survey paper compares different hashing algorithms and their drawbacks. The user must know the types of attacks and must apply appropriate hashing algorithm to avoid attacks.

REFERENCES

1. Review Paper on Secure Hashing Algorithm and Its Variants – By Priyanka Vadhera, Bhumika Lall (http://www.ijsr.net/archive/v3i6/MDIwMTQxOA==.pdf)

2. Cryptographic Hash Functions: A Review – by Rajeev Sobti,G.Geetha(http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.7241&rep=rep1&type=pdf)

3. https://en.wikipedia.org/wiki/Cryptographic_hash_function

4. SHA-1 hash function under pressure – heise Security

5. Henri Gilbert, Helena Handschuh: Security Analysis of SHA-256 and Sisters. Selected Areas in cryptography2003

6. http://www.unixwiz.net/techtips/iguide-cryptohashes.html

7. https://media.blackhat.com/us-13/US-13-Aumasson-Password-Hashing-the-Future-is-Now-WP.pdf

8. http://stackoverflow.com/questions/326699/difference-between-hashing-a-password-and-encrypting-it