Yuliia STEPANENKO. Valeriia SOLODOVNIK
Scientific supervisors: Andrii FESENKO, Larysa MURYTENKO

SECURE PASSWORD STORAGE WITH CRYPTOGRAPHIC HASH FUNCTION

Summary: The analysis of methods of user authorization in the system is carried out. The password hashing algorithm is analyzed. Comparison of popular hashing algorithms is carried out.

Keywords: password, hash algorithm, authentication, password cracking, deterministic function, iterations, salt.

1. Formulation of the problem

Passwords play a critical role in our whole life. It's the most common security method to authenticate or verify a user's online identity. They provide a powerful guard against unauthorized access to systems and data, and are ubiquitously used in various online activities such as communication, learning and, of course, shopping and banking.
Unfortunately, passwords suffer from two seemingly intractable problems: password cracking and password theft. Password cracking (also known as password guessing) is an attack in which an adversary attempts to guess the users' password. For security reasons, it makes sense to store passwords in hashed form. This guards against the possibility that someone who gains unauthorized access to the database can retrieve the passwords of every user in the system.
Hashing performs a one-way transformation on a password, turning the password into another String, called the hashed password. «One-way» means that it is practically impossible to go the other way – to turn the hashed password back into the original password (Fig. 1).

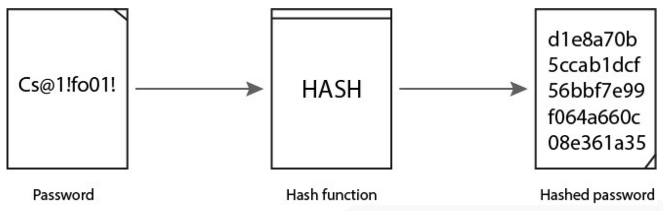

Fig. 1. Hashing algorithm flow example – one-way

There are numerous functions used for password hashing including: MD5, SHA1, SHA256 - SHA512, PBKDF2, BCRYPT, SCRYPT and Argon2.

2. Conclusion

Various types of cryptographic systems exist that have different strengths and weaknesses. Typically, they are divided into two classes; those that are strong, but slow to run and those that are quick, but less secure.
According to Jeff Atwood, "hashes, when used for security, need to be slow." A cryptographic hash function used for password hashing needs to be slow to compute because a rapidly computed algorithm could make brute-force attacks more feasible, especially with the rapidly evolving power of modern hardware. We can achieve this by making the hash calculation slow by using a lot of internal iterations or by making the calculation memory intensive. It's important to achieve a good balance of speed and usability for hashing functions. A well-intended user won't have a noticeable performance impact when trying a single valid login.
MD5, SHA1, SHA256, SHA512 hash functions can be executed in parallel on multi-processor systems, fact that increases significantly the efficiency of password guessing attacks. MD5 and SHA-1 are common hashing algorithms used today but it's a necessary to avoid these algorithms because these technologies are deprecated. There are many examples of hashes that are appropriate for use when storing passwords, such as PBKDF2, Argon2, Scrypt, or Bcrypt.

Ministerstwo Edukacji i Nauki
Dofinansowano z programu „Doskonała nauka" Ministra Edukacji i Nauki