SALTING VS STRETCHING PASSWORDS FOR ENTERPRISE SECURITY

Password security is important. Salting and stretching a password are two ways to make passwords more secure from attackers. You can use the strategies separately or deploy them in tandem for the most security.

This article covers the logic of password protection, including salting and stretching passwords.

Database attacks

Because passwords are a user's key to their data, they are a key target for attackers.

Popular methods of password attacks happen through brute force and rainbow attacks:

- **Brute force attacks** are a trial and error approach where a computer guesses until it gets it right. This may sound ineffective, but computers can try many, many times. In this era of computing, "Hashcat breaks an 8 character full coverage (a-zA-Z0-9!-=) password in 26 days on a single 1080 Nvidia GPU."
- **Rainbow attacks** are another form of cracking passwords, where all possible combinations of hashed passwords are pre-computed and stored in a dictionary. Then, an attacker runs through their list of hashes to find a match. If their scan returns a match, then they have the password.

Passwords help prevent attacks

First things first: nothing online is perfectly secure. Not even computers not connected to the internet are perfectly secure. We use passwords to minimize risk of attack, not to guarantee it will never happen.

Though you cannot guarantee security, there are some ways to increase database security. Salting and stretching passwords are two such strategies:

1. **Salting passwords.** Designing the password encryption so only one password is compromised rather than the whole database. Attackers will attack. Don't make it easy for them to run off with the whole loot at once.
2. **Stretching passwords.** Lengthening the password (on the database side) so the time it takes to crack the password becomes too expensive for attackers. The idea is that attackers will opt for easier targets—makes common sense, similar to the rumor that crime rates fall proportionately to how high one travels up the hills in San Francisco.

Ways to store passwords

To understand password salting and stretching, let's look at ways companies can store their data.

It is critical to note: Responsibility does not fall on the company's shoulders if an individual user compromises their own password. A company can encourage a user to use stronger passwords by enforcing character limits and special character sequences. A company, however, cannot control if a user allows someone walking by to see their password.

A company's responsibility is to *secure* their stored passwords.

Direct password storage

Storing the password as-is is the worst possible way to store passwords in a database. If a person with no computer received the list, they can read the whole list of passwords as they are.

Password	Database
123456	123456
123456789	123456789
qwerty	qwerty
password	password

Simple hash function

A better, but far from perfect, option is to apply a hash function to the password and store the hash value in the database. This is an added step between the phrase the user inputs and the phase (hash) that ultimately gets stored in the database.

Password	Hash Function	Database (Hex MD5 Hash)
123456	Hash = H(password)	8531ab28b1ffc32016b5f38e7f650f7b
123456789	Hash = H(password)	4b9ff53081aee2b193e85a007c5bdf34
qwerty	Hash = H(password)	c45a108d730a41f40ff525b5a3b039bb
password	Hash = H(password)	0c6975129201c9956a91428a952923c4

Here's how it works: If an attacker were to receive or obtain the list of passwords (the right-most column, above), but the passwords are hashed, the values are unreadable. Therefore, a computer would have to figure out what function was used to turn those values into the original password.

Hashing sounds good, but it is an all-or-nothing proposition: If an attacker were to crack the hash function, then the hacker could read all the passwords in the database.

Salting a password

This is where salting comes in. A salt adds a string of characters to the user's passwords to just before the password undergoes hashing. The salt accomplishes two things:

Attackers cannot do a dictionary lookup to see how popular passwords get hashed. Because there is a random string of values added to the password, passwords no longer exist as "popular strings" and are more random. Their complexity has increased greatly.

A unique salt per user prevents an attacker from guessing the hash function *and* unlocking an entire database of passwords. The added step between the password and the hash function makes it so if an attacker gets the hash function figured out, they still have to run through many more combinations to guess the unique salt value.

Password	Salt	Hash Function	Database (Hex MD5 Hash)
123456	6d 4d 90 9b 18 5c 28 7e	Hash = H(password + salt)	81fd6c878635663a461c47f5b4ec7b19
123456789	20 f3 35 86 98 d8 a2 95	Hash = H(password + salt)	7d17f9a7a6cdf9786644c1677f8cd5cb
qwerty	e1 86 e4 b2 49 e5 24 bb	Hash = H(password + salt)	958a562333317cd430e23683d21fdaaf
password	54 0a 45 8b f6 65 92 fb	Hash = H(password + salt)	2ab60f7bd474ceaf133248e28b80cb49

Stretching a password

Finally, a tried-and-true method to frustrate attackers is stretching passwords before they're saved to the database. The primary aim of stretching a password is to make deciphering the password more costly—whether with memory, time, or money—than an attacker can afford.

In stretching, the strength of a password is measured by its bits of key strength. Methods of lengthening the number of bits a password has comes down to the hash function. Usually, hash functions are looped thousands of times, simulating randomness and adding more and more bits to the complexity of a password passed to the database.

Password	Salt	Hash Function (10,000 loops)	Database (Hex MD5 Hash)
123456	6d 4d 90 9b 18 5c 28 7e	Hash = H-1000(password + salt)	0486bd80c7bff90b3c087571f28c515104e9f3bca43b41d0cc875fcb2acfbab78cd68caddb776a2f8112fb76e2a2c374585e8634bc0a97ff305eb9704daf2e90
123456789	20 f3 35 86 98 d8 a2 95	Hash = H-1000(password + salt)	561621c4a2832a326688a8db188159db9200ce8e3d87009f9decd6cd95f40e6571db20bc6b479c223eb742e3f5778b1c69ddc8cde3c57902f1ec2fb2cb803d00
qwerty	e1 86 e4 b2 49 e5 24 bb	Hash = H-1000(password + salt)	69d3b54b1af944358e43e66465632a467c2e0f4d8a385178efbcf8a09ce436c409722a17e1ed05ec2e2c9e6b58a93a1f23fcd5809b6ced0517ec000d87119700
password	54 0a 45 8b f6 65 92 fb	Hash = H-1000(password + salt)	c6e1be4ef759ef0e5e32ed455555eaf39c253b616e346d804f308ec622583f307e526930757478b6fda3d7451e9e85170bac7c9cb145f9c0be5581ac0936ad5f

How to stretch passwords

There are popular, well-developed tools one can use to start stretching passwords, including PBKDF2 and Bcrypt. Today, 128-bit keys are a common benchmark for effective password encryption.

For the developer, the goal of password stretching software is to increase computational time on the attacker's system. Stretching maximizes the difficulty an attacker may have to decrypt the data, while still maintaining usability of the application itself.

When a password is encrypted, the user has to wait for their password to run through the hoops and get verified against their actual password. If it takes the user's computer 3 minutes to hash their password and check it against the database, that might be unreasonable. But if the user's password is submitted and verified with the database in a few milliseconds, then there could be room for improvement.

For companies, perhaps the best improvement is to limit the instances of user passwords.

Additional resources

For more on cybersecurity topics and practices, browse our BMC Security & Compliance Blog or check out these articles:

- Cybersecurity: A Beginner's Guide

- What Is a Cyber Resilience Strategy?
- Introduction to Enterprise Security
- 5 Examples of Recent Data Breaches