Passwords

Passwords play a crucial and significant role in virtually every individual's life. People employ passwords to access their personal services and data, safeguarding information they wish to keep private and exclusive to themselves. To keep data safe and private, user has to take password implementation seriously. There are numerous ways and technologies of enhancing passwords to ensure privacy. A poorly implemented password exponentially increases the risk of getting hacked and losing private and sensitive data.

A password's strenght is measured by a consept called entropy. The higher the entropy, the stronger and more secure the password is. For example, adding uppercase letters to a password that contains 8 lowercase characters significantly increases the password's entropy.

Measuring the entropy of a randomly generated password is much easier than measuring the entropy of a human-generated password. Measuring entropy in the context of human-generated passwords typically disregards human behavior, making it more difficult to measure entropy correctly. Moreover, accounting for human behavior becomes challenging due to the substantial variation observed among individuals, further complicating the accurate assessment of entropy in passwords. Humans often exhibit predictable patterns by incorporating common words and names into their password creation, lowering its entropy significantly.

Besides following obvious patterns, humans often go for basic passwords like "password123," making their choices even more guessable. This compounds the problem since these passwords are not just simple and pattern-based but also commonly used alongside other easily guessed ones. Humans also tend to disregard important aspects of passwords, for much more meaningless features. For example, Humans often prioritize memorability over complexity when choosing passwords, leaving them significantly more vulnerable to brute force attacks.

Even though human-generated passwords generally have lower entropy than randomly generated ones, various techniques are applied to enhance their security after creation. For instance, processes such as hashing and salting are commonly employed to make these passwords significantly more challenging to crack. Salting passwords is commonly done before the hashing process. Salt in the context of passwords means adding a unique value to the end of the user's password, making the hash stronger and also adding complexity to the password itself making it harder to hack. A salt could for example be a random string that is added to the end of the password, like ”abcd”. Each salt is always unique to the user's password.

When a password is created, it is commonly stored in a database of the service that the password was created for. In order to protect the passwords in the database, a technique called hashing is implemented to each password before storing them to the database. Hashing is the process of transforming given data in to a seemingly random string of characters. Hashing enables the storing of passwords in a secured format to the database, while keeping the actual plaintext password safe by not storing it anywhere. The same string always produces the same hash, enabling the authentication process of services. When a user logs in, the entered password is hashed and compared to the hashed passwords stored in the database. If the entered password matches any of the stored hashes, authentication is successful.

Hashing is done by hash functions. Hash functions that are considered strong and reliant are available in the internet and are commonly used by many when hashing passwords. For example the argon2 along with bcrypt are commonly used hash functions. Hashes are considered irreversible, making the attackers attempt of decryption nearly impossible. For user experience it is more efficient to use bcrypt, since it is faster. Despite bcrypt being the more suitable for lower user experience delay, it doesn't have all the capabilities of argon2. For example, argon2 is more resistant to attacks due to its design to be memory-hard.

Even though hashing is considered security proof, there are ways to crack hashes. For example, brute forcing hashes is possible and also using a rainbow table attack. A common defense against rainbow table attack is adding salt to each password. Key streching algorithms are common when hashing passwords, since they make the brute forcing and attacking process of the attacker much more harder and time consuming. Key streching algorithms make the process of hashing longer adding computational effort to make it more difficult for the attacker to crack the password.

no system is never completely secure and human behaviour despite all the techniques of securing passwords, can lead to vulnerabilities and security risks. There are public services available to ensure that user's account hasn't been hacked. It is important to use these services to ensure that user's account hasn't been hacked, since it may not always be clear and the attack can happen under the user's blind eye.

Multi factor authentication enhances security and reduces the risk of getting hacked significantly. Multi factor authentication adds another factor to the authentication process on top of the password. For example the user might have to confirm the authentication through a phone app after providing the password. So even if an attacker manages gain access to the user's password, they would still have to confirm the authentication through the victim's phone.

Well crafted passwords are typically hard to remember for humans, which brings out the dilemma of usability and security in the context of passwords. Long enough, unique and strong passwords are crucial for security, but they are hard to remember, which can lead to humans choosing weak password or reusing old passwords. Reusing old passwords enables the attacker to acces not only the account he targeted, but also the victims other accounts from different services. Password managers are considered to be a good solution to the dilemma of usability and security, since they generate and store passwords for the user, enabling more complex and stronger passwords, since the user technically doesn't have to remember them.