Passwords summary

zaterdag 7 januari 2023 13:01

Online attacks: repeated guessing, trying to login. Server can deny access after repeated failure.
Offline attacks: repeated guessing, checking for correctness. Check must be made expensive.

Attacks: interception, social engineering

Password generation schemes:

- Names, birthdays, other 'security questions'-like answers (easy to guess, ***especially offline***)
- Pick word from dictionary (still easy to guess offline)
- Obfuscated word (i.e. apply transformations to word) (still easy to guess offline)
- Diceware (several random words) (fairly expensive to guess, but still easy to enumerate using dictionary)
- Derive password from passphrase (e.g. replace words in phrase by their first character)
- Randomly generated strings (hard to remember)
- Derive passwords from a master password using a key derivation function.

Some remaining problems/advices:

- Password reuse
- Mandatory password changes (can lead to worse security)
- Insecure backup systems (e.g. security questions to restore a password can be less secure than the password itself)
- Use two-factor authentication

Note that passwords (or at least: their hashes) can be obtained in data breaches.

Password hashing

When password are hashed using the same function, the same password returns the same hash. This allows for pre-computing hashes to save time. Main workaround: use a salt, and then compute the hash of the concatenation of the salt with the password.

Hash chain: apply a hash function multiple times in a row (with a function in between to ensure the output is in the domain of the hash function once again)

PBKDF2 (Password-Based Key Derivation Function 2) is a function which uses a pseudorandom function (e.g. HMAC-SHA1), a password and a salt to derive a key. Its downsides include depending on a hash function designed for speed, and not using much memory.

Bcrypt is a function which uses Blowfish encryption, together with a cost, salt and password to derive a key. It has a cost factor to account for changes in computing power, but still does not require much memory.

scrypt is a function which has one cost factor for both time and memory. It's based on PBKDF2, but relatively new. An attack on it is based on a time-memory trade-off, since its memory requirements are based on backreferences to earlier outputs in a hash chain.

Time-space trade-offs in password hashing attacks enable the use of GPUs/ASICs (see quiz)

Argon2 is a function which first expands to the entire available memory, then applies a sequence of memory-hard transformations, and concludes by absorbing the entire state into a small tag. It allows for separate parameters for time and memory cost.