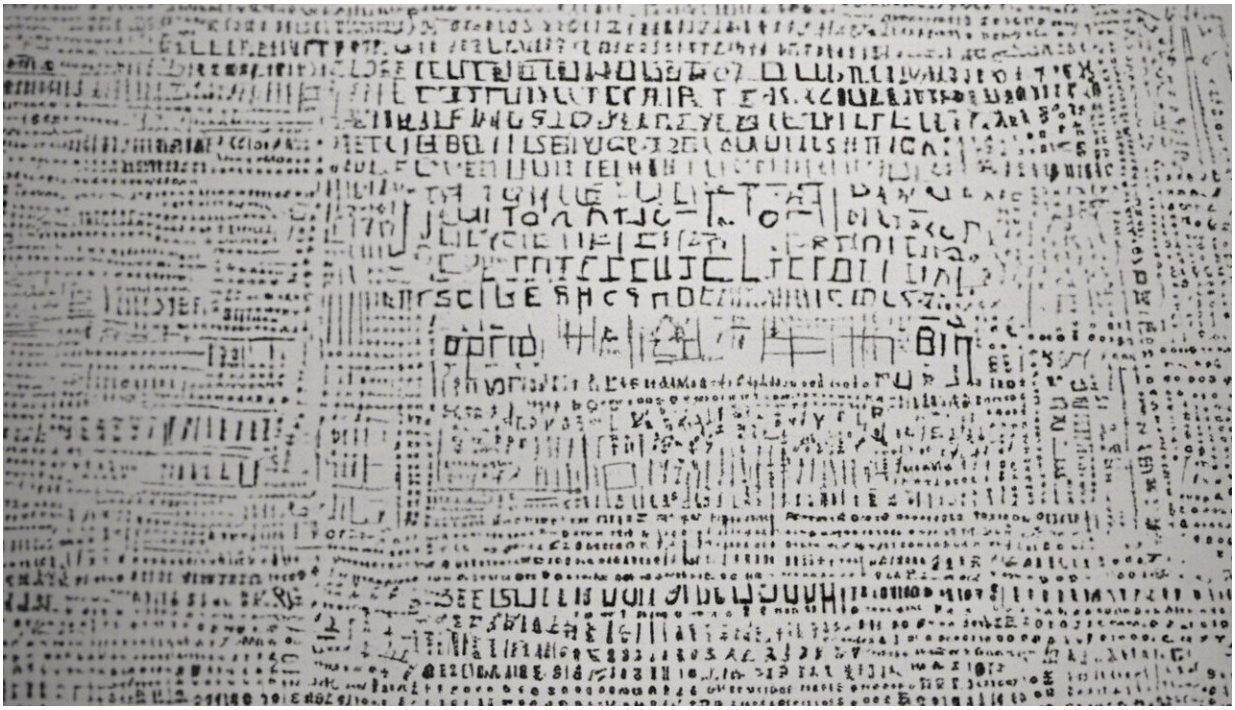
Making and breaking passwords

March 24 2016, by Mihai Lazarescu, Sciencenetwork Wa



Credit: AI-generated image (disclaimer)

The issue of choosing a good password is still key for cyber security because users tend to pick passwords that are easy to remember rather than secure. So what are the key attributes of a good password?

First, one needs to consider the length. The longer the password (or rather passphrase), the more difficult it is to crack it.

Why? Because when hackers try to crack passwords, they cannot tell how close they are to guessing it.

For example, a hacker cannot tell if they managed to get 10 out of 12 characters right. There are only two outcomes from each try—it either works or it does not.

Ideally, when creating a password someone would select four or five words and combine them to form a strong password.

For example, if one likes Star Wars, they can use the character names to build a strong password: LukeSkywalkerDarthVaderDarthSidious (length=35).

Second, one needs to have largest characters set possible if they can only have a limited password length.

If you restrict your set to only letter and digits you already reduce the complexity of the password.

Ideally, you should use letters (upper and lower case), digits, special characters (such as ^ or }), punctuation signs and the commonly overlooked empty space.

Why do this? Because most password cracking tools using dictionaries and variations of words (such big, biG, Big, Big1, Big11 and so on) rarely check for variations such as big{}8==.

There is always a chance that someone will try to brute force check all combination of characters (including special characters and punctuation signs) but the time it takes to go through all the combinations makes it less likely that they will crack it before you change your password again.

Third, the frequency with which the password is changed is critical. If the length of the password is short (say 12-16), then you want to change it more frequently.

So what if one needs to have passwords for multiple accounts? Assuming that length is not an issue, use a passphrase and change one word that would be easy associate with the account.

For example, using the base passphrase

LukeSkyWalkerDarthVaderDarthMaul, one can specialise it for two accounts say office LukeSkyWalkerOfficeDarthVaderDarthMaul and banking LukeSkyWalkerDarthVaderBankingDarthMaul.

All of this effort is designed to make it as difficult as possible for hackers to crack passwords, so how do hackers actually crack passwords?

On most systems, passwords are stored as hashes—the hash is generated in most cases using a well-known cryptographic hash function which maps the input data sequence of any length into a finite data sequence.

The cryptographic hash function is irreversible meaning that the input sequence cannot be rebuilt from the hash.

The problem is that the hashes generated are not necessarily unique for each input sequence—two different input sequences may have (depending on the hash function) the same hash and it is this weakness that the hackers target.

The actual password cracking process is effectively finding an input sequence that has the same hash as the target password.

To crack the password, the hackers use automated tools and to speed up the search, hackers will also use dictionaries and hash tables.

Most people will invariably set their password based on something meaningful and thus words (and their variations) are prime candidates to be tried.

Hash tables help significantly when the hacker has obtained a list of hashes and is dealing with "unsalted" passwords as the processing associated with generating the hashes has already been done.

Some operating systems such as MS Windows use "unsalted" passwords and thus hash tables can be used.

Other operating systems such as Linux try to make the process of password cracking more time consuming by adding another string called a "salt" to the password before generating the hash.

For example, if the unsalted password is LuzernGenevaBaselZurich then that phrase would be converted via a hash function.

However, if the password is salted, then a "salt" string is added to the start or end of the password and then the hashing is carried out.

So if the salt is 18967802 then the computer stores the password as the hash of 18967802LuzernGenevaBaselZurich.

The benefit of using a "salt" is that it renders hash tables useless and makes the guessing process a lot more time consuming and difficult for hackers.

This article first appeared on ScienceNetwork Western Australia *a science news website based at Scitech.*

Provided by Science Network WA