Hashing

What Is It?

An extension of cryptography, hashing is the process of mapping data to a sequence of fixed length. Where this also differs from encryption, is that hashing is a one-way algorithm, i.e., the hashing algorithm cannot be reversed to get the original text/information. Therefore, this is mostly used as a method of verification, for example with passwords. An organisation does not need to know each of their users' passwords, so they simply store the hash, and each time the user attempts to login again, it compares the hash of their input to the hash they already have stored. Nowadays with advanced technology, we consider 256 bits the minimum length of the hash (the output of whatever mapping algorithm used) required to be secure.

How Can It Be Exploited?

Dictionary Attacks:

We are always told not to use common passwords like 'qwerty' or '123456', because they can easily be guessed. In your head, you might envision a man in all-black clothing and a balaclava sitting at a computer pondering over common passwords, "hmm maybe I should try 'hello' next", but it is in fact *much* easier and more sinister than that. People will create a list of the most used passwords (and more - SecLists) and all the attacker has to do is run a script with a couple of lines or use a popular tool like Hashcat to try all the possible options. This is known as a Dictionary Attack. To give you an idea, I've used Hashcat in the past and it took about 5 minutes to check 10000 passwords that used a bcrypt hash, and I think that extends to about 1000000 in 5 hours. While that may seem like a long time, imagine you were a motivated attacker, and it only took you *half a day* to gain access to your target. This is why it is so important that you use a complex password with a combination of letters and numbers, because even if your password gets hashed, it doesn't protect you against a simple dictionary attack.

Rainbow Tables:

The main idea behind hashing is that it protects users against a data leak on the organisation's side, as a jumble of letters and numbers is useless to an attacker, but it may not be as secure as you think. If the hashing algorithm is fairly weak, an attacker may be able to use a **Rainbow Table**, which is a precomputed table that contains the unencrypted text of various hashes.

They can simply compare your hash to the one in their Rainbow table. Although there are several prevention methods (including just using a stronger hash), a direct counter to Rainbow Tables is using a 'salted hash', which involves combining the hashed password with a random value, meaning that even two identical passwords would have different hashes, making it much harder to crack.

Birthday Attack:

A Birthday Attack involves trying to find two messages that have the same hash i.e., find a hash collision. They take advantage of the property that hashes are always of a fixed length, and hence there must be a finite number of hashes. The attack gets its name from the Birthday Problem, which states that in a group of 23 people there is a 50% chance that at least two people have the same birthday. In fact, that

increases to 97% in a group of 50 people. This same concept applies to hashes, and attackers try to find two messages with the same hash output. If they can do this, it is possible for them to maliciously tamper with data or bypass a security system.

Chosen-plaintext Attack:

When the attacker has access to use the (hidden) hash function, they are able to input carefully chosen plaintext messages and analyse their hashed output. This may allow them to determine the actual hashing algorithm used, allowing them to crack any hashed message.