High-Performance Applications

Xenia Bogomolec

1 Introduction

The increasing performance of technologies always comes with a side effect: hacking tools become more efficient as well. There are several open-source tools publicly available, which benefit from parallel computing as well as efficient operations. We want to introduce you to one of them, the password recovery tool hashcat (https://hashcat.net/hashcat/).

2 Hash Functions

Password recovery is just a nice term for hacking hashed passwords. Since passwords need to be known to their owner only, it is a basic security requirement to only store hashes of passwords on the server side. Ideally, the application provider should not be able to recover a password. This is ensured by so-called hashing algorithms, which produce a pseudo random output (pseudo random generators). That means that the hash value is a random number (in hexadecimal representation) or random string of fixed length, regardless of the length of the input string. Unlike an encryption algorithm,

X. Bogomolec (✉)
Quant-X Security & Coding GmbH, Hanover, Germany
e-mail: xb@quant-x-sec.com

V. Liermann and C. Stegmann (eds.), *The Digital Journey of Banking and Insurance, Volume II*, https://doi.org/10.1007/978-3-030-78829-2_10

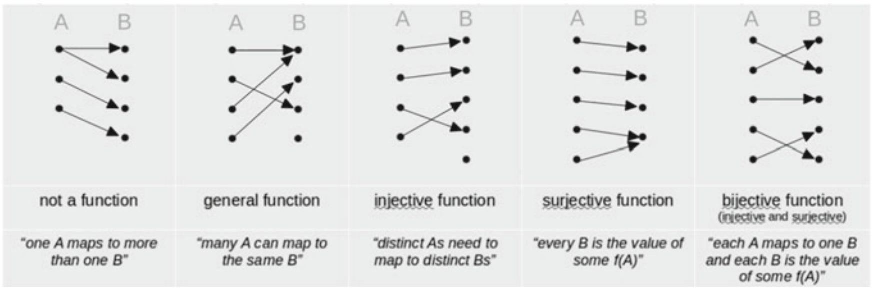

A B
not a function
"one A maps to more than one B"
general function
"many A can map to the same B"
injective function
"distinct As need to map to distinct Bs"
surjective function
"every B is the value of some f(A)"
bijective function
(injective and surjective)
"each A maps to one B and each B is the value of some f(A)"

Fig. 1 Math functions (© Quant-X Security & Coding GmbH)

hashing algorithms cannot be inverted. In mathematical terms, hash functions are not even injective: there are an infinite number of inputs but only $2^{(\text{hash length in bits})}$ possible outputs. This basic condition makes a backward map from hash value to input string impossible. The incredible security of hash functions stems from the property of collision resistance: finding two inputs with the same hash must be impractical, i.e. taking far too much time to compute (Fig. 1).

The original password of a given password hash can only be recovered by testing all possible passwords (the password keyspace during an attack) until a match is found. The simplest approach to this is called brute force attack. For the hashing and encryption algorithms currently used, this can take up to billions of years, depending on available technologies and knowledge. Mask attacks take advantage of how people can remember passwords easily. In the worst case, such a password recovery only takes a few seconds.

3 Password Cracking

First, the attacker needs to gain access to the hash of a password. This can happen, for example, through a malicious insider with administrator rights or any kind of a data breach. Countless huge providers have been affected by such breaches; Equifax is just one big name among them. The so-called hashing "salt", a random string which is added to the password to be hashed, does not increase the security of the hashed password itself. It is naturally stored with the password hash. Otherwise, the stored hash cannot be compared to the hash of the password during a login process. Usually the salt gets lost in a data breach together with the related password. The true benefit of the salt is: even if a password is used via multiple platforms or by several users, its hash will look different in each provider's database or in a password hash list published by hackers.

Once the hash is known, the amount of possible password candidates can often be reduced considerably with additional information, such as the known maximum length and allowed characters of a password. Such information can easily be found on user registration pages. Password hash lists from data breaches usually contain the hashing algorithm, user name, the password hash and the salt.

3.1 Attack Mechanisms

There are two main hash cracking patterns: brute force attacks and mask attacks.

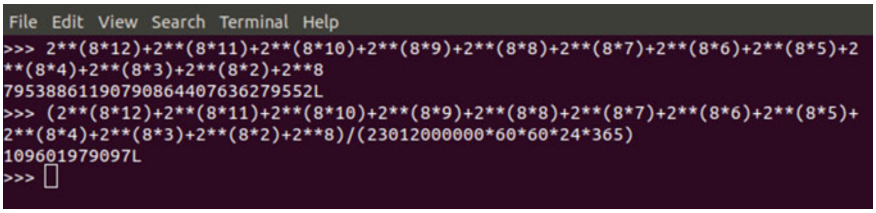
3.1.1 Brute Force Attack

A **brute force attack** consists of computing the hashes of all possible password candidates with the known salt and comparing them with the password hash—until a match is found. Traditionally, this attack starts with all possible 1-character strings, then all possible 2-character strings and so on. For our example, we chose SHA-2, a hashing algorithm which is still widely used for password hashing.

If the password length n is not known and any possible 8-bit byte is allowed as a character, the maximum number of iterations (hash function calls followed by comparisons of the password candidate hash against the original password hash) will be $2^8 + 2^{16} + 2^{24} + \ldots + 2^{8n}$ for a password with n characters. This is the maximum because only as many of the 2^{8n} password candidates in the last term will have to be tested until the right password candidate is found. Given the above situation, a brute force attack on a password with 12 characters consists of

$$\text{maximum}\, 2^8 + 2^{16} + 2^{24} + \cdots + 2^{96} = 79,538,861,190,790,864,407,636,279,552$$
$$\text{minimum}\, 2^8 + 2^{16} + 2^{24} + \cdots + 2^{88} = 310,698,676,526,526,814,092,329,312$$

iterations on password candidates. With a rate of 23,012 million hashes per second for SHA-2 on an 8x Nvidia GTX 1080 processor, the attack takes more than 109,601,979,097 years to complete in the worst case, and about 428,132,730 years in the ideal case. The ideal case would be to find the original password with the first of the 2^{96} 12-character iterations (Fig. 2).



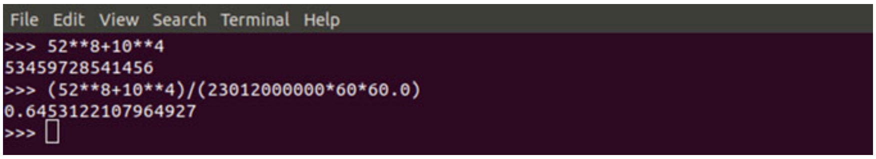
```
>>> 2**(8*12)+2**(8*11)+2**(8*10)+2**(8*9)+2**(8*8)+2**(8*7)+2**(8*6)+2**(8*5)+2
**(8*4)+2**(8*3)+2**(8*2)+2**8
79538861190790864407636279552L
>>> (2**(8*12)+2**(8*11)+2**(8*10)+2**(8*9)+2**(8*8)+2**(8*7)+2**(8*6)+2**(8*5)+
2**(8*4)+2**(8*3)+2**(8*2)+2**8)/(23012000000*60*60*24*365)
109601979097L
>>> 
```

Fig. 2 Minimum iterations and required number of years (original screenshot) (© Quant-X Security & Coding GmbH)

Hashcat offers automated and configurable brute force attacks. For example, a restricted character set, minimum and maximum password candidate length can reduce the password candidate keyspace for an attack considerably.

3.1.2 Mask Attacks

Mask attacks are more sophisticated. Here, the password candidate keyspace is reduced considerably by applying a popular character pattern to all inputs. An example of such a pattern is a name and year. Thus, we fix the last 4 characters as digits {0,.., 9} and all previous characters as upper or lowercase letters {A, …, Z, a, …, z}. If the password length is 12, the necessary mask attack iterations reduce to a maximum of $52^8 + 10^4 = 53{,}459{,}728{,}541{,}456$ password candidates. With the same cracking rate of 23,012 million hashes per second, it takes less than 40 minutes to find the original password of the user (Fig. 3).



```
>>> 52**8+10**4
53459728541456
>>> (52**8+10**4)/(23012000000*60*60.0)
0.6453122107964927
>>> 
```

Fig. 3 Maximum iterations and required number of years for mask attack (original screenshot) (© Quant-X Security & Coding GmbH)

3.2 Benefit of Slow Hash Functions

The length of SHA-2 hashes is 256 bits, so each comparison of SHA-2 hashes equals the comparison of 256 bits. If you look at hashcat password cracking speed lists on the internet (e.g. https://gist.github.com/epixoip/a83d38f412b4737e99bbef804a270c40), you will find that the cracking of SHA-2 hashes is only about three times slower than the cracking of SHA-1 hashes. This fact holds in spite of the 2^{128} times higher cryptographical complexity of SHA-2 over SHA-1 as functions. As a conclusion: SHA-2 hashed passwords are only about three times safer against brute force attacks with the tool hashcat than SHA-1 hashed passwords—a difference of no concern to a hacker.

The tool hashcat achieved such high performance by taking advantage of two facts:

1. Bit arrays, an array data structure that compactly stores bits. This structure is the basis for the bitmap tables, which are created specifically for each attack.
2. Parallel computing of hashes on available GPU (graphical devices) or CPU processing units.

Leveraging parallel computing depends on the algorithm. Some candidates do not offer this possibility, e.g. the winner of the Password Hashing Competition in July 2015, Argon2. Argon2 offers three configurable versions suitable for specific circumstances. One configuration parameter is the degree of parallelism. Argon 2 is designed by Alex Biryukov, Daniel Dinu and Dmitry Khovratovic from the University of Luxembourg.

The best protection against password cracking is to use slow hashing algorithms. The hash cracking speed for bcrypt on an 8x Nvidia GTX 1080 processor, for example, is 105,700 hashes per second. The above mask attack would take 16 years instead of 40 minutes if bcrypt was the chosen password hashing algorithm.

4 Summary

Anyone who knows how to use the password cracking tool hashcat can decrypt a poorly chosen password encrypted by a fast hashing algorithm, such as the ones from the SHA family, on their own personal computer in quite a short time. The price of an 8x Nvidia GTX 1080 processor on Amazon is currently around €300. Believing that you will never be affected by a data breach is not a safe attitude anymore. Neither can the complexity of the password requirements outpace the speed of evolving technologies. Slow hashing algorithms for password storage are the right answer to ever-growing cybercrime intelligence.