

CYBERSECURITY LAB #2



Giacomo Gori – Tutor
didattico

g.gori@unibo.it

Exercise



Complete the exercises, taking notes of all the steps that you take



Write a **small** report and upload it on Virtuale



Remember: write name, surname and the number of the lab session on the report!

Prerequisites

Virtualbox and the configured
Kali VM.

Instructions are on Virtuale!



BY OFFENSIVE SECURITY

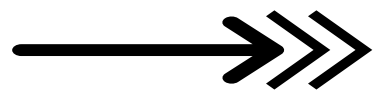
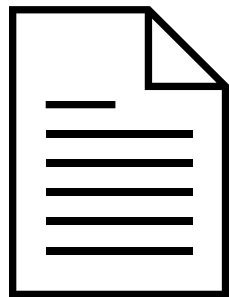
Password



What is a HASH?

It works as a **fingerprint**:

- It's a cryptographic function that create a unique code from items (text, file, ...) of different lengths



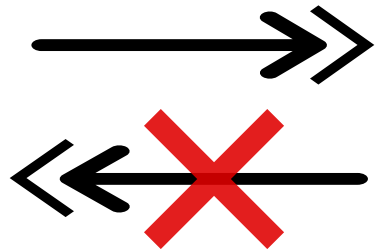
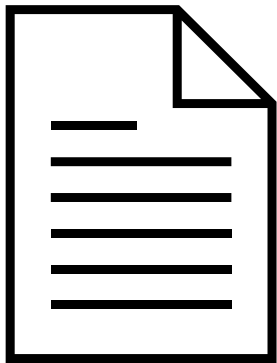
fe6e32b7af5.....



Characteristics of hash functions

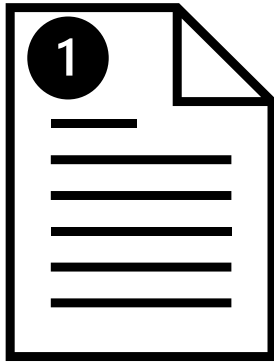
- **Quick** to calculate
- **Small** size of the output
- A small change in the input generate a **big change** in the output
- Weak and strong **resistance to collisions**
 - i.e. it's impossible to find two input that have the same hash as output
- **Irreversibility**

Irreversibility?

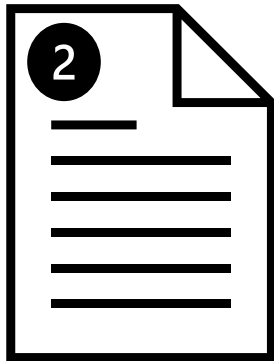


fe6e32b7af5.....

Resistance to collisions?



fe6e32b7af5.....





Is it even possible?

Goals of hash functions

Integrity

- Be sure that a file is not (un)wittingly modified during transmission or when stored

Confidentiality

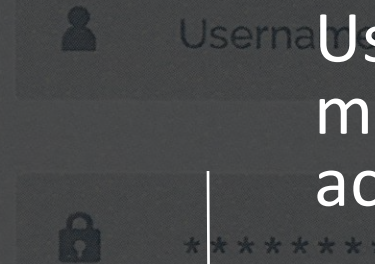
- Instead of saving password in plaintext, save only the hash!
 - This way, if a dataleak happens, attackers will not know the original passwords of users..



"Oh no, they hacked my Instagram account!"

But wait, did they really
hacked a more than \$100
billion company just to
steal your account?

It's more likely that they did a **password reuse attack**



Username

Users often set the same password for multiple services (e.g. Instagram, bank account, ..)

If the password used for a service gets leaked, it can be used to access other accounts of the same users

There's a whole business model behind this

- Attackers do not use user data themselves, but they mostly **sell them on the dark web**

Hash functions available

MD5

128 bit

SHA-1

160 bit

SHA-2
family

(224,256,384,512
bit)

And much more..

Less famous or insecure

GNU coreutils

- They are tools available in most Linux distributions
- Basic usage
 - Generate hash: ***{md5, sha1, sha224, sha256, ...}sum (filename)***
 - Check hash, example with md5: ***echo "(hash) (filename)" | md5sum -c***
 - There are 2 spaces between hash and filename!!

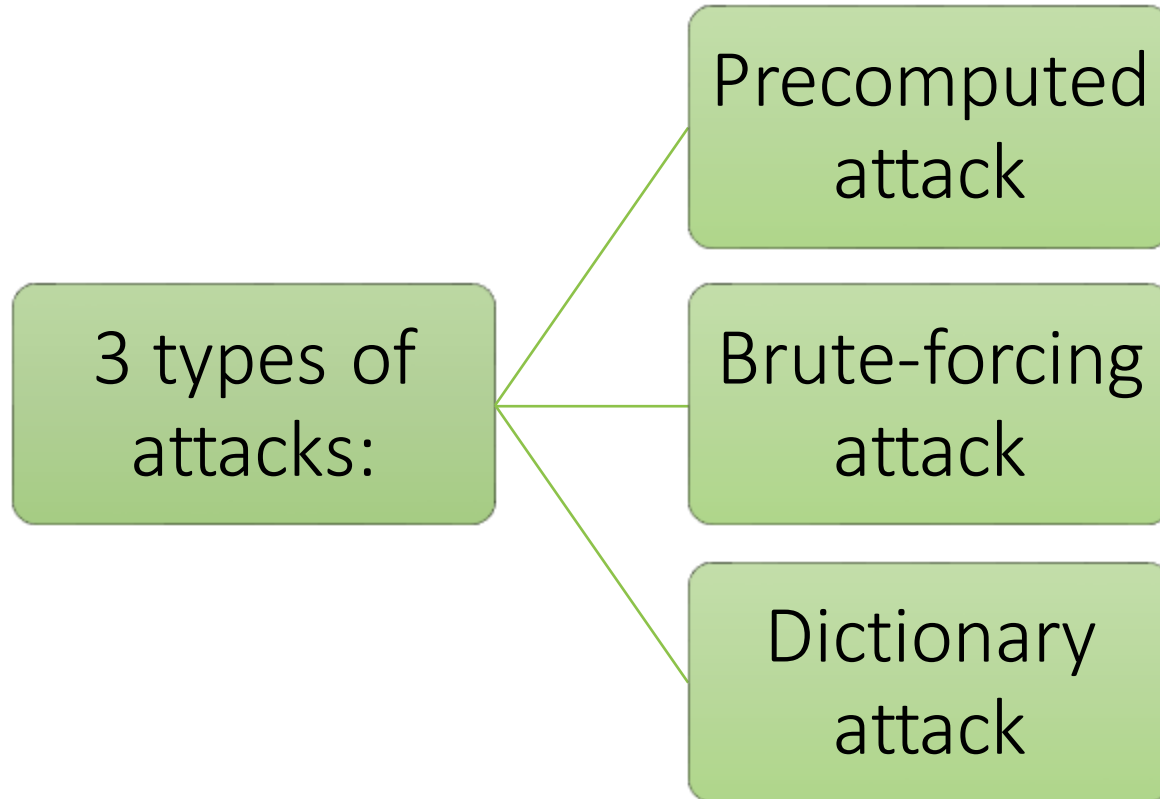
Breaking hash algorithms

The goal: find the plaintext from the hash, even if it should be irreversible.

In fact, we can break only insecure hash algorithms...

Anyway, we will focus not on the inner workings of each algorithm, instead we will **try to find the plaintext** using different techniques

Attacking hash algorithms



The goal: obtaining the plaintext, *trying* with different combinations

Precomputed attacks

- Space/time tradeoff
 - Save time precomputing hashed of the most common (or likely) passwords, but take up space by storing them
- Smarter method: **rainbowtables**
 - More sophisticated but require less storage



Rainbowtables

Install rainbowcrack

- *sudo apt install rainbowcrack*

Generate and sort a MD5 rainbowtable

- *sudo rtgen md5 loweralpha 1 7 0 1000
100000 0*
- *sudo rtsort /usr/share/rainbowcrack*

Crack a given hash

- *rccrack /usr/share/rainbowcrack -h (hash)*

Try to crack
them

HASH n1:

*6e6bc4e49dd477ebc
98ef4046c067b5f*

HASH n2:

*e879410167dfb867
0e483f7f7a1843cf*

Defending
against
precomputed
attacks





Salting

Adding some random piece of data (**salt**) to the passwords

Storing that piece in plaintext, together with the password

Precomputation becomes useless

Salting example

ID	PASSWORD (HASH)	SALT
1	6e6bc4e49dd477ebc98ef4046c067b5f	7b6b1c1077c3fbe74b19
2	7a36be31fbe72d24c2d4bdb44c8055a6	41942cad1e6c17e7e2e3
3	0225205578734fc6ea670eae72e92160	32f38b5e593075f974d7
4	a00a34a06520ccf4d7e0e3d6442cb85f	e6dde301236a3891ca88
5	3ba94ed6ae8ac1891ef96c136853a5cc	2ff137a14978890fe1e9
6	e20d4b60cd5a8ebe1ca51b52eb0a1377	f9ee4a12e3ea69aa7be0

It's not the end
of the story...

If hackers have the hash of a password from a dataleak, typically they also have the salt, **because it's stored in plain and in the same database!**

So, Dictionary and Brute-forcing attacks can still be performed.

Brute-force

- Just **try every possible combination** until you find the right one
 - Could be an «infinite» process
- With long password could take forever!
 - Masks could be useful



Dictionary attacks



- **Try all the words in a predefined list**
 - Smarter lists are the best option!
 - That's why the strongest passwords are the ones more "randomly" made
- In a real world scenario, weeks or even months may be needed!
 - Hash cracking can be optimized by running many instances in parallel, so using more cores (e.g. GPU, FPGA, ...)

Dictionary attacks

- As said, smarter wordlist are the best option
 - **People choose common words as password**
 - Wordlists are made available online
 - Some of them created using actual credentials from public leaked databases
 - You can also generate your own lists
- In kali you have a built-in wordlist
 - Extract it: `sudo gunzip /usr/share/wordlists/rockyou.txt.gz`



Most common passwords

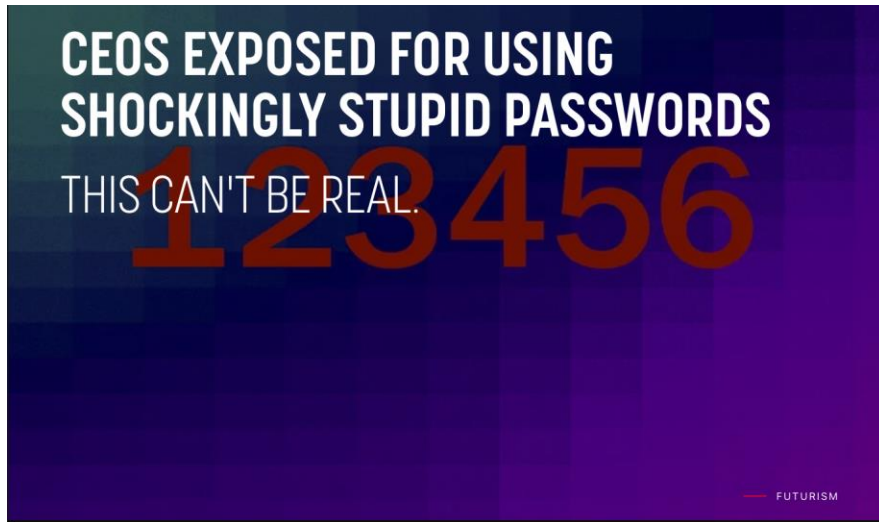
- 
- 1.password
 - 2.123456
 - 3.123456789
 - 4.guest
 - 5.qwerty
 - 6.12345678
 - 7.111111
 - 8.12345
 - 9.col123456
 - 10.123123
 - 11.1234567
 - 12.1234
 - 13.1234567890
 - 14.000000
 - 15.555555
 - 16.666666
 - 17.123321
 - 18.654321
 - 19.7777777
 - 20.123

And in Italy?

- | | |
|--------------|----------------------|
| 1. 123456 | 11. giulia |
| 2. 123456789 | 12. 1234 |
| 3. password | 13. amoremio |
| 4. ciao | 14. <i>CENSURATA</i> |
| 5. juventus | 15. francesca |
| 6. napoli | 16. francesco |
| 7. ciaociao | 17. 1234567890 |
| 8. 12345 | 18. alessia |
| 9. 12345678 | 19. qwerty |
| 10. martina | 20. andrea |



With no exceptions...



Dutch prosecutors say Trump's Twitter account was hacked by guessing his password: maga2020!

This is not a joke. Trump's old Twitter password was "maga2020!"

By Alex Ward | @AlexWardVox | alex.ward@vox.com | Dec 16, 2020, 3:50pm EST



Listen to this article

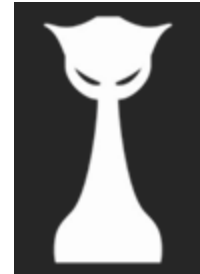


Password123

According to [new research](#) from password manager NordPass, countless high-level executives and CEOs are using mind-meltingly terrible passwords, including — we are not joking — "123456" and "password."

Cracking password

- **First step:** we need to understand from a hash which algorithm was used
 - The **hashid** command can help, but not always
- **Second step:** try to find the right combination
 - We will see the hashcat command
 - Usage: **hashcat (-m mode) (-a attack) (hash) [OPTIONS]**



Hashcat

- Usage: **hashcat** (-m mode) (-a attack) (hash) [OPTIONS]
 - **Mode:** choosing the algorithm (es: 0 for MD5, 100 for SHA1, ..)
 - **Attack:** dictionary, brute force, using masks (es: 0 dictionary, 3 bruteforce, ..)
 - **Hash:** a string or a file containing one or more hashes
 - **In OPTIONS:** can be introduced the wordlist
- Cracked hashes are saved in the "potfile" in ~/ .hashcat/hashcat.potfile
 - Use --show to compare the input hashlist with the potfile, showing the cracked ones (--left, for the opposite)

Hashcat

- The **man** command will show you the various configuration. Some of them:

#	Name	Category
900	MD4	Raw Hash
0	MD5	Raw Hash
100	SHA1	Raw Hash
1300	SHA2-224	Raw Hash
1400	SHA2-256	Raw Hash
10800	SHA2-384	Raw Hash
1700	SHA2-512	Raw Hash
17300	SHA3-224	Raw Hash
17400	SHA3-256	Raw Hash
17500	SHA3-384	Raw Hash
17600	SHA3-512	Raw Hash
10	md5(\$pass.\$salt)	Raw Hash salted and/or iterated
20	md5(\$salt.\$pass)	Raw Hash salted and/or iterated
110	sha1(\$pass.\$salt)	Raw Hash salted and/or iterated
120	sha1(\$salt.\$pass)	Raw Hash salted and/or iterated

Example

- **Create the MD5 hash of the word «hola»**

- *echo -n "hola" | md5sum*
- Risultato: *4d186321c1a7f0f354b297e8914ab240*
- In this case, the hashid answer is ambiguous: MD2/4/5?

- **Crack it with hashcat**

- So, MD5 means **m = 0**, dictionary attack is **a = 0**, let's use the wordlist rockyou.txt
- The command is:

```
hashcat -a 0 -m 0 "4d186321c1a7f0f354b297e8914ab240"  
/usr/share/wordlists/rockyou.txt
```

Example with salt

- **Let's reuse the previous word («hola»), but let's add the salt**
 - **Salt:** 1234
 - **Hash with salt:** ccee5504c9d889922b101124e9e43b71

- **Crack it with hashcat**

- The syntax is *hash:salt*
- The command is:

```
hashcat -a 0 -m 10 "ccee5504c9d889922b101124e9e43b71:1234"  
/usr/share/wordlists/rockyou.txt
```

Try to crack it

HASH:

bc107137cda7aa074
de2664a88247f2dfa5
4546923049ec92986
9edd6bc648a0

SALT:

dd1b1n5

Masks in hashcat



- **You can also be smarter with brute force attacks**
 - For example, a lot of passwords are name and birth year, mine would be Giacomo98 :)
 - What if I say to hashcat to try only passwords with a predefined structure?
- **Masks are the solution (-m parameter)**
 - You can say to hashcat to try only words with a particular pattern



Rules in hashcat

- **But rules are even more powerful**
 - Change the wordlist trying to follow some pattern
- You can create your own rules
 - E.g. \$x to append 'x' at the end of every word
- Hashcat has some built in rules in `/usr/share/hashcat/rules/`
 - Use rules with the `-r` parameter

Example with rules

- **Let's use a new word: «Hola123!»**

- **Hash:** *401518eee35b49f00bc0a3ab74c4915e*
- This word is not included in rockyou.txt, so hashcat wouldn't be able to crack it without rules (i.e. changing the «hola» word)

- **Crack it with hashcat and rules**

- Let's use an example rule from the hashcat folder
- The command is:

```
hashcat -a 0 -m 0 -r /usr/share/hashcat/rules/T0XICv2.rule  
"401518eee35b49f00bc0a3ab74c4915e" /usr/share/wordlists/rockyou.txt
```

Try to crack it

HASH:

0e8ae09ae169926a
26b031c18c01bafa

HINT: It contains a
phrase without spaces
and some numbers at
the end

Try to crack it

HASH:

c73fceaab80035a7
5ba3fd415ecb2735

HINT: It contains, in
order: a common word,
some numbers and a
special character

Exercise



Crack the 5 hashes with rainbowtables or *hashcat*, taking notes of all the steps that you take



Write the report, showing the commands and why you choosed them, together with the cracked passwords



Remember: write name, surname and the number of the lab session on the report!