CYBERSECURITY LAB #2

Jocelyne Elias
Giulio Tripi

Exercises

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!

3 weeks Deadline

You can do the exercise and the report now, or later at home

Every project must be submitted within 3 weeks from the day of the practice exercise.

Ethical Hacking

What does "HACKER" mean?

"A **hacker** is a person skilled in information technology who **uses their technical knowledge to achieve a goal or overcome an obstacle**, within a computerized system by non-standard means. "

Source: Wikipedia

It depends on the side that you choose....

White hat

Black hat

Ethical Hackers (aka White hat)

They help companies, organizations and developers to check and improve their security.

With **bug bounty programs** OR being "hired" by them, performing **VAPT**

hackerone

bugcrowd

Bug Bounty programs

Offering a reward to hackers that find undisclosed security bugs

An example

Apple Security Bounty program

Products	Topic	Reward Range	View Examples
Device attack via physical access	Lock Screen bypass	$5,000 – $100,000	
	User data extraction	$5,000 – $250,000	
Device attack via user-installed app	Unauthorized access to sensitive data	$5,000 – $100,000	
	Elevation of privilege	$5,000 – $150,000	
Network attack with user interaction	One-click unauthorized access to sensitive data	$5,000 – $150,000	
	One-click with elevation of privilege	$5,000 – $250,000	
Network attack without user interaction	Zero-click radio to kernel with physical proximity	$5,000 – $500,000	
	Zero-click unauthorized access to sensitive data	$5,000 – $500,000	
	Zero-click kernel code execution with persistence and kernel PAC bypass	$100,000 – $1,000,000	

Sell the vulns

Zerodium Exploit Acquisition Program

ZERODIUM Payouts for Desktops/Servers*

Windows
macOS
Linux/BSD
Any OS

RCE: Remote Code Execution
LPE: Local Privilege Escalation
SBX: Sandbox Escape or Bypass
VME: Virtual Machine Escape

Up to $1,000,000										1.001 Win RCE Zero Click (Win)
Up to $500,000								3.001 Chrome RCE+LPE (Win)	2.001 Apache RCE (Linux)	2.002 MS IIS RCE (Win)
Up to $250,000						5.001 MS Outlook RCE (Win)	4.001 MS Exchange RCE (Win)	2.003 OpenSSL RCE (Linux)	2.004 PHP RCE (Linux)	
Up to $200,000	6.001 VMware ESXi VME	5.002 Thunderbird RCE (Win/Linux)			4.002 Sendmail RCE (Linux)	4.003 Postfix RCE (Linux)	4.004 Dovecot RCE (Linux)	4.005 Exim RCE (Linux)	2.005 nginx RCE (Linux)	
Up to $100,000		3.002 Safari RCE+LPE (Mac)	3.003 Edge RCE+LPE (Win)	3.004 Firefox RCE+LPE (Win)	5.003 Word/Excel RCE (Win)	7.001 WordPress RCE (Linux)	7.002 cPanel/WHM RCE (Linux)	7.003 Plesk RCE (Linux)	7.004 Webmin RCE (Linux)	
Up to $80,000	6.002 VMware WS VME (Win/Linux)					5.004 Adobe PDF RCE+SBX (Win)	5.005 WinRAR RCE (Win)	5.006 7-Zip RCE (Win)	6.003 Windows LPE/SBX (Win)	
Up to $50,000	6.004 USB LPE (Win/Mac)	8.001 Antivirus RCE (Win)			5.007 WinZip RCE (Win)	5.008 tar RCE (Linux)	6.005 macOS LPE/SBX (Mac)	6.006 Linux LPE (Linux)	6.007 BSD LPE (BSD)	
Up to $10,000	9.001 Routers RCE	8.002 Antivirus LPE (Win)	7.005 phpBB RCE (Linux)	7.006 vBulletin RCE (Linux)	7.007 MyBB RCE (Linux)	7.008 Joomla RCE (Linux)	7.009 Drupal RCE (Linux)	7.010 Roundcube RCE (Linux)	7.011 Horde RCE (Linux)	

** All payouts are subject to change or cancellation without notice. All trademarks are the property of their respective owners.*

2019/01 © zerodium.com

Sell the vulns

Zerodium Exploit Acquisition Program

ZERODIUM Payouts for Mobiles*

FCP: Full Chain with Persistence
RCE: Remote Code Execution
LPE: Local Privilege Escalation
SBX: Sandbox Escape or Bypass

iOS
Android
Any OS

Up to $2,500,000										1.001 Android FCP Zero Click (Android)
Up to $2,000,000										1.002 iOS FCP Zero Click (iOS)
Up to $1,500,000									2.001 WhatsApp RCE+LPE Zero Click (iOS/Android)	2.002 iMessage RCE+LPE Zero Click (iOS)
Up to $1,000,000									2.003 WhatsApp RCE+LPE (iOS/Android)	2.004 SMS/MMS RCE+LPE (iOS/Android)
Up to $500,000	3.001 Persistence (iOS)	2.005 WeChat RCE+LPE (iOS/Android)	2.006 iMessage RCE+LPE (iOS)	2.007 FB Messenger RCE+LPE (iOS/Android)	2.008 Signal RCE+LPE (iOS/Android)	2.009 Telegram RCE+LPE (iOS/Android)	2.010 Email App RCE+LPE (iOS/Android)	4.001 Chrome RCE+LPE (Android)	4.002 Safari RCE+LPE (iOS)	
Up to $200,000	5.001 Baseband RCE+LPE (iOS/Android)		6.001 LPE to Kernel/Root (iOS/Android)	2.011 Media Files RCE+LPE (iOS/Android)	2.012 Documents RCE+LPE (iOS/Android)	4.003 SBX for Chrome (Android)	4.004 Chrome RCE w/o SBX (Android)	4.005 SBX for Safari (iOS)	4.006 Safari RCE w/o SBX (iOS)	
Up to $100,000	7.001 Code Signing Bypass (iOS/Android)	5.002 WiFi RCE (iOS/Android)	5.003 RCE via MitM (iOS/Android)	6.002 LPE to System (Android)	8.001 Information Disclosure (iOS/Android)	8.002 [k]ASLR Bypass (iOS/Android)	9.001 PIN Bypass (Android)	9.002 Passcode Bypass (iOS)	9.003 Touch ID Bypass (iOS)	

** All payouts are subject to change or cancellation without notice. All trademarks are the property of their respective owners.*

VAPTs

Performing Vulnerability Assessment and Penetration Testing requires some phases:

Scope and plan → Information gathering → Vulnerability analysis → Exploitation → Reporting

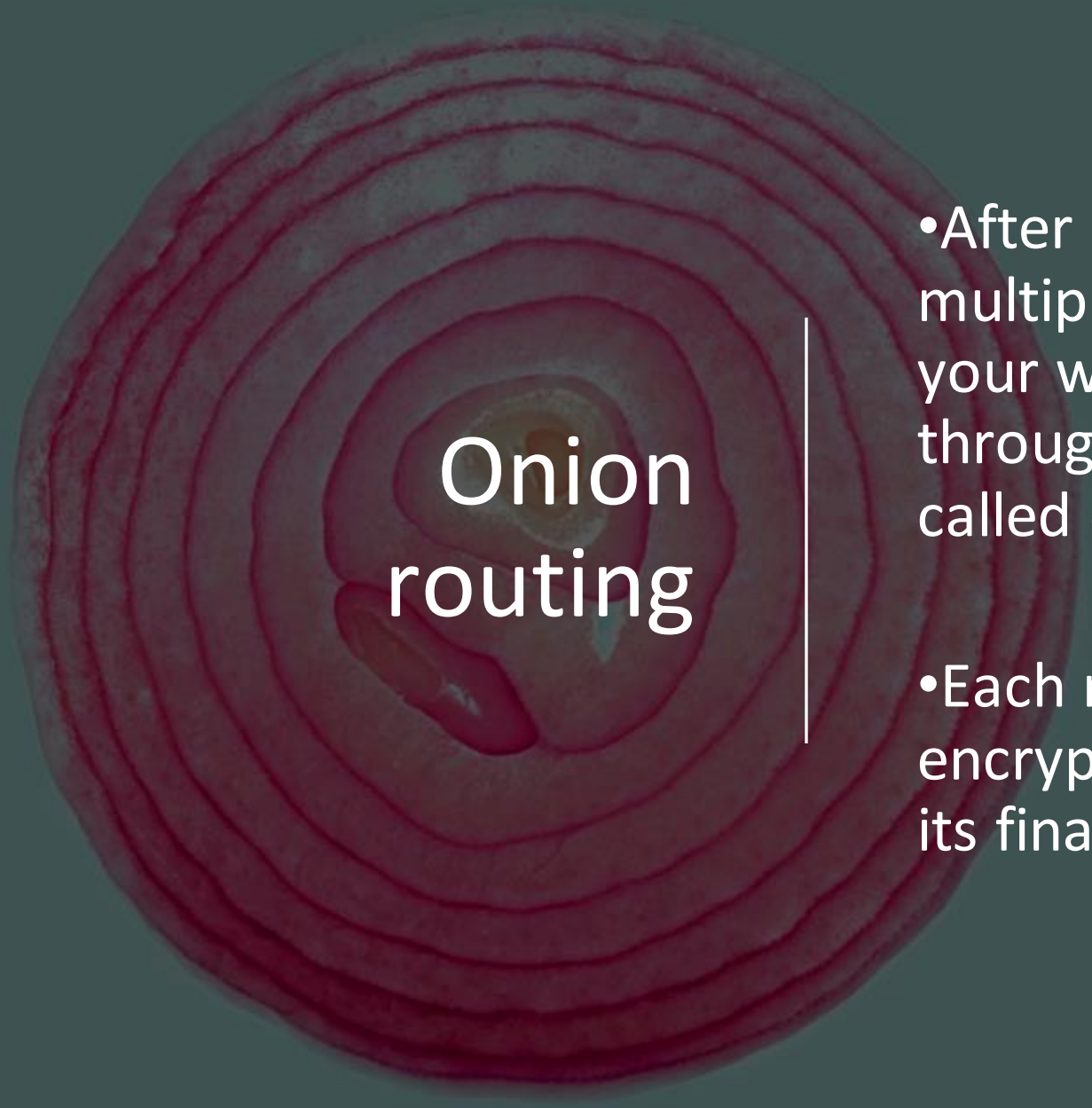
Getting experience..

- Trying to attack deliberately vulnerable VMs
- Participating to Capture The Flag (CTF)

CYBER CHALLENGE.IT

- But **NEVER TRY WITH REAL TARGETS!**
  - Unless you have the authorizations :)

Hidden services



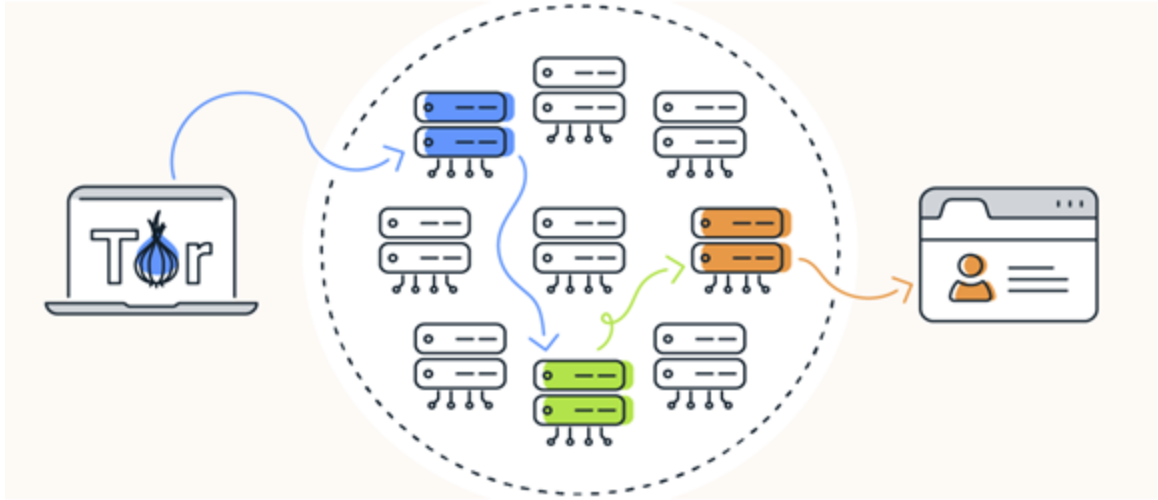
Onion routing

- After your data is secured inside multiple layers of encryption, your web traffic is transmitted through a series of network nodes, called onion nodes.

- Each node “peels away” a layer of encryption until the data reaches its final destination, fully decrypted.

TOR browser

Tor is a browser that anonymously transmits encrypted data across three layers (*entry – middle – exit nodes*) of international proxies that make up the Tor circuit.



Nodes

Here user data is fully decrypted, being sent through a series of nodes which decrypt your data one layer at a time.

To ensure anonymity, **each middle node knows only the identity of the preceding and the subsequent middle nodes**, without knowing who is the initiator or destination.

Am I anonymous with TOR?

TOR anonymity

TOR can hide your IP address and browsing activity using the **multi-layered encryption,** but there's no such thing as perfect online anonymity.

Moreover, you still can be identified if you log in to an online account or provide details to a website.

Hidden services

Tor hidden services work within the Tor network and allow you to register an internal Tor-only service that gets its own **.onion hostname**.

Tor resolves those .onion addresses and directs you to the service hidden behind that name.

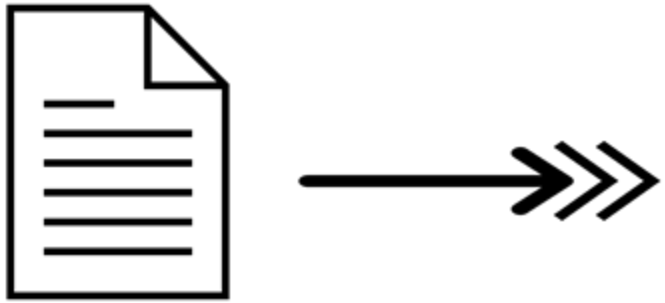
Hidden services provide **two-way anonymity**: the server doesn't know the IP of the client and neither the client knows the IP of the server.


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

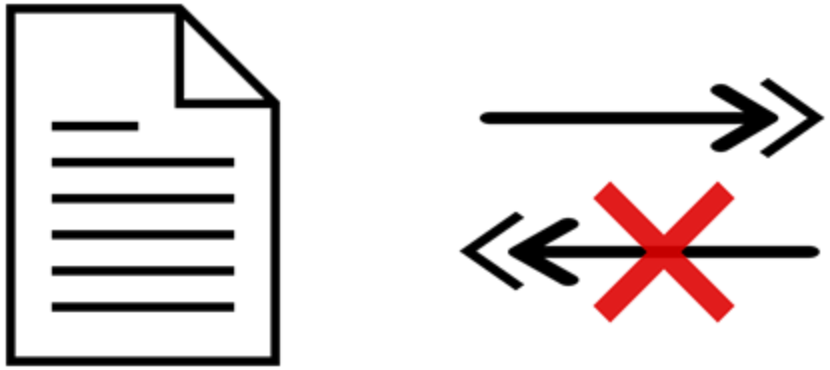


fe6e32b7af5.....

Characteristics of hash functions

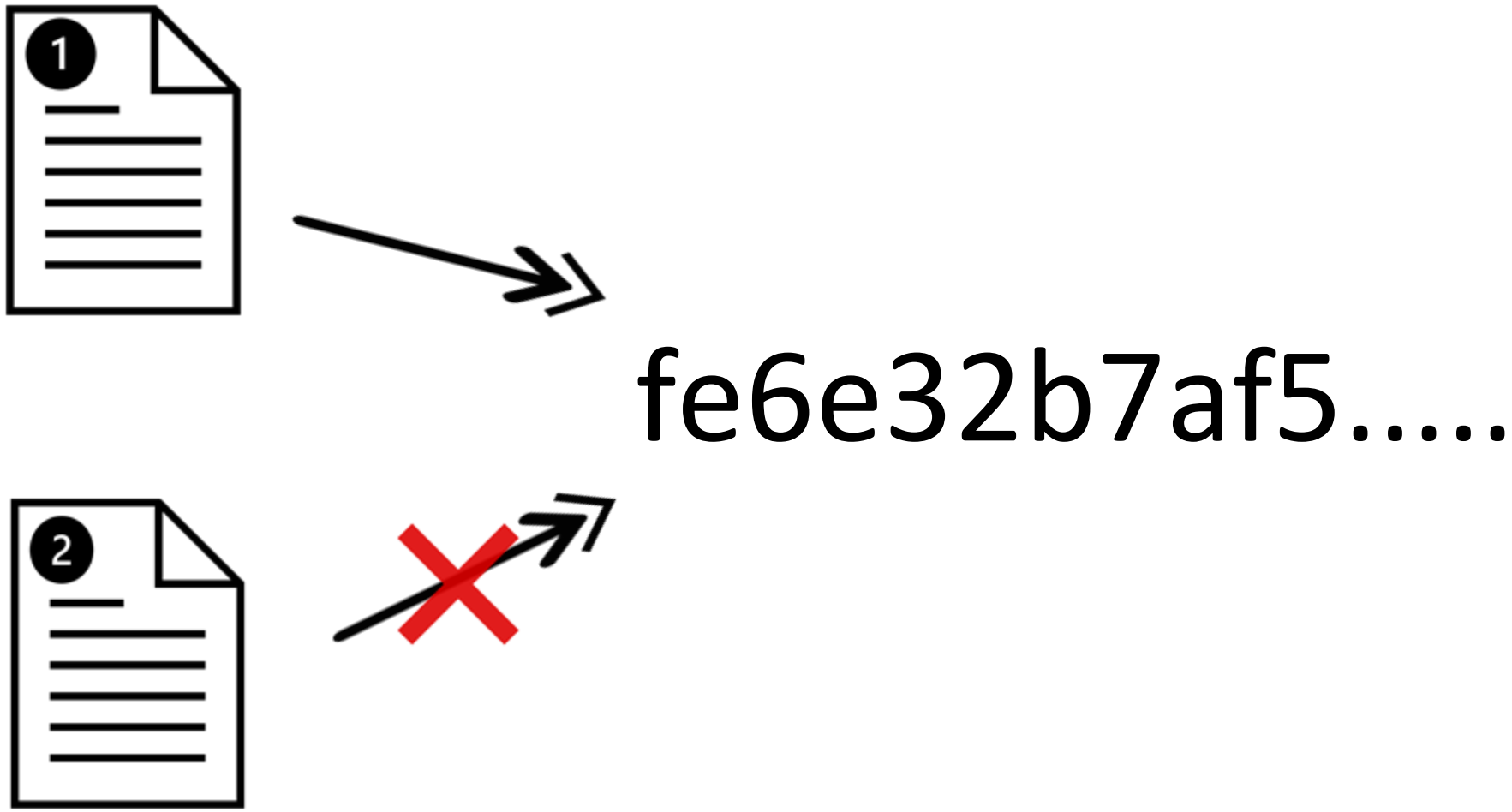
- **Quick** to calculate
- **Small** size of the output
- A small change in the input generate a **big change** in the output
- Weak and strong **resistance to collisions**
  - i.e. it's impossible to find two input that have the same hash as output
- **Irreversibility**

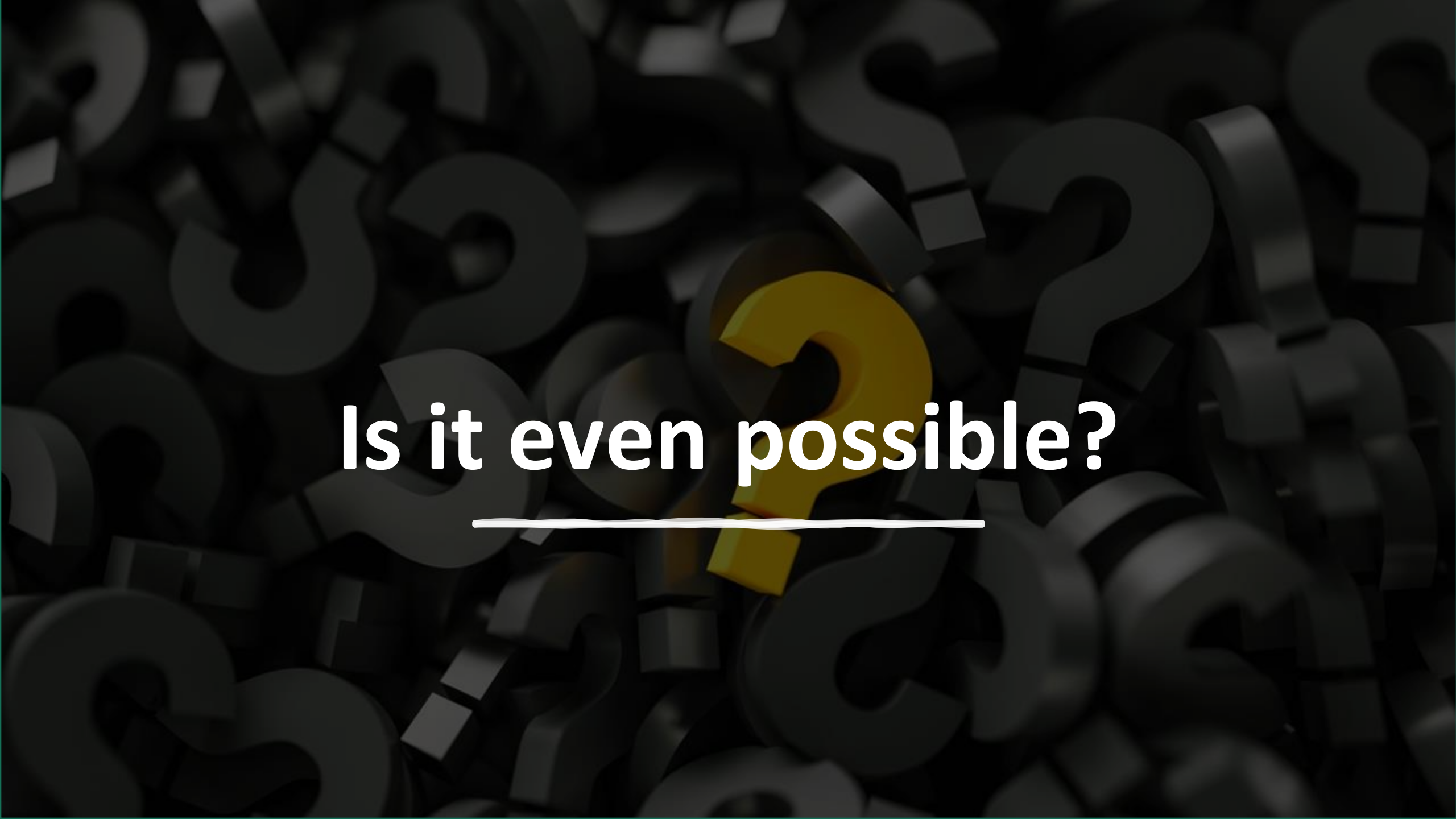
Irreversibility?



fe6e32b7af5.....

Resistance to collisions?





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 Instragram 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**

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

3 types of attacks:

- Precomputed attack
- Brute-forcing attack
- Dictionary attack

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

- rcrack /usr/share/rainbowcrack -h (hash)

Try to crack them

HASH n1:

6e6bc4e49dd477eb c98ef4046c067b5f

HASH n2:

427ade9c15ec6437 51860eba9899355b

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 dabatase**!

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*

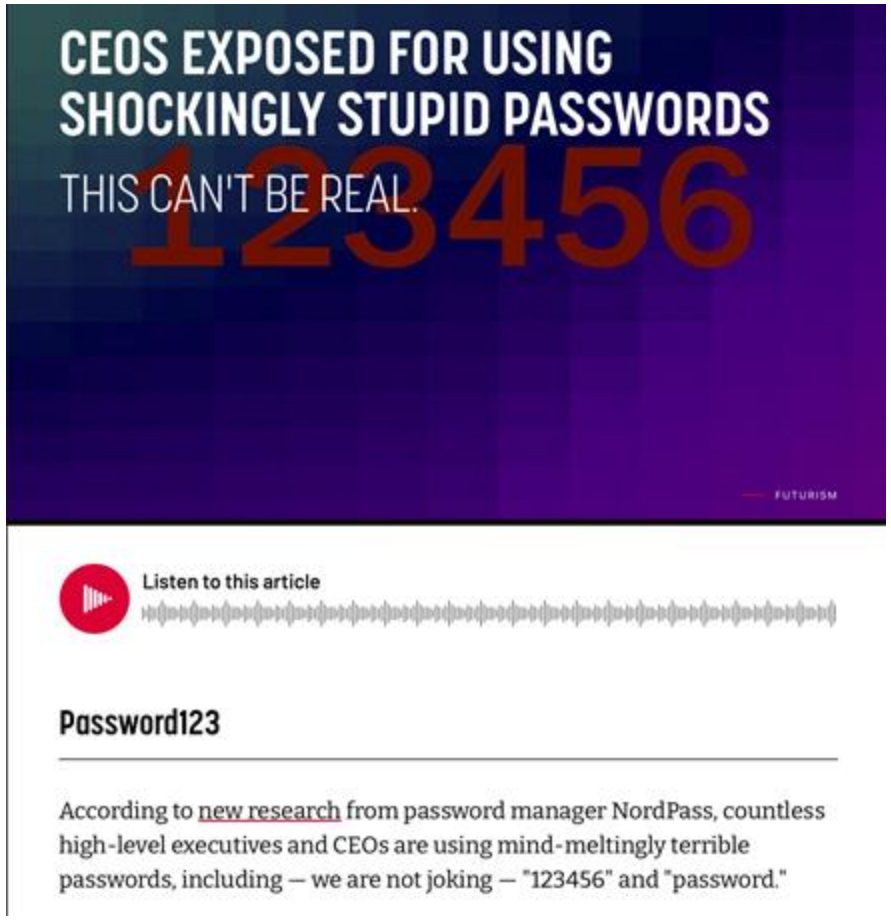
Most common passwords

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
2. 123456789
3. password
4. ciao
5. juventus
6. napoli
7. ciaociao
8. 12345
9. 12345678
10. martina
11. giulia
12. 1234
13. amoremio
14. *CENSURATA*
15. francesca
16. francesco
17. 1234567890
18. alessia
19. qwerty
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

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 ambigous: 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 sintax is *hash:salt*
  - The command is:

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

Try to crack it

HASH:

6c00f2d6e1610bfc9b415daf80d45855f2c56443c2dc2f71e7ef27168d1f2857d6168f4d374ed8eca349f2debd18d4ccac339218ca70446adf99906039574 2b4

SALT:

hjt88q

Masks in hashcat

- **You can also be smarter with brute force attacks**
  - For example, a lot of password are name and birth year, mine would be Giulio98 :)
  - What if I say to hashcat to try only password 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/T0XlCv2.rule "401518eee35b49f00bc0a3ab74c4915e" /usr/share/wordlists/rockyou.txt
  - https://hashcat.net/wiki/doku.php?id=rule_based_attack
  - Other ruleset: InsidePro-PasswordsPro.rule

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

Try to crack it

HASH:
dc612dc12fb4540a88b88875c2bee3b4

HINT: It contains, in order: a common word and 1 or 2 numbers. The common word has the case iNVERTED

Exercise

Crack the 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!

Part 2: overthewire.org

- It's a site with a lots of challenges to do, helping you to learn and practice security concepts in the form of games.

- **Objective of this lesson: Try *Bandit* levels, from Level 0 to 10.**

Tips

- The exercises on the website will suggest you the command to use, together with some useful link on the web
- The "*man*" command is your friend :)

- **Try to resolve them on your own!!**

Connecting with ssh

Use the *ssh* command to connect, with user bandit0 (0 stand for the first level):

```
ssh bandit0@bandit.labs.overthewire.org -p 2220
```

Exercise 2 : www.overthewire.org

Complete the levels, from 0 to 10, taking notes of all the steps

Write a small report

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