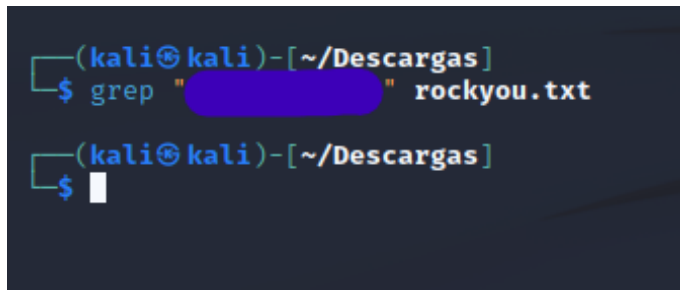


Romper contraseñas

1º: busca si tu contraseña esta en rockyou

Podemos ver que efectivamente, mi contraseña no esta en el diccionario rockyou

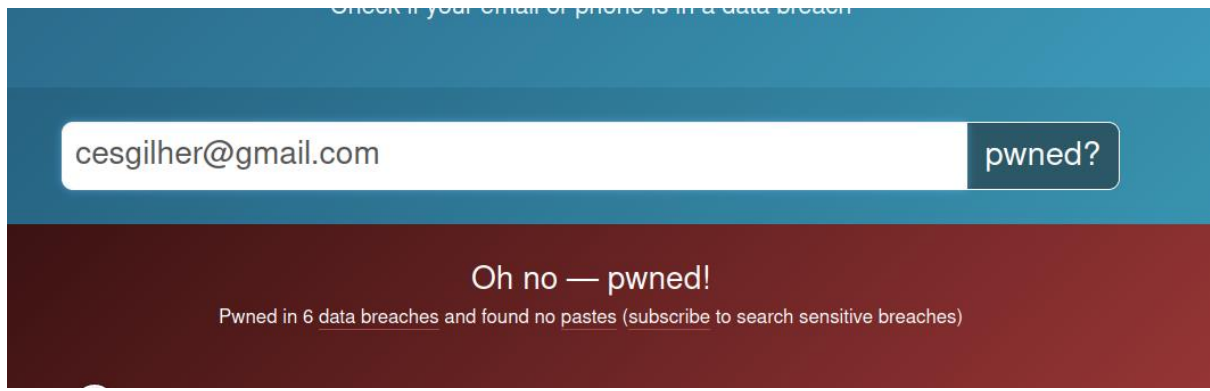


```
(kali㉿kali)-[~/Descargas]
$ grep "cesargilher" rockyou.txt

(kali㉿kali)-[~/Descargas]
$
```

2º: buscar si habeis sido pawneados

Accedemos a la web <https://haveibeenpwned.com> e introducimos nuestro correo electronico.



Lo adecuado aquí seria acceder a las paginas afectadas y cambiar la contraseña.

3º: buscar el algoritmo hash mas seguro de estos HMAC-SHA512,PBKDF2-HMAC-SHA256,bcrypt,LM,NT

Al final el algoritmo mas seguro es aquel que mas cuesta deshasear, y eso se puede averiguar con la cantidad de contraseñas que se analizan por segundo. Siguiendo esta logica el algoritmo mas seguro es el bcrypt.

```
G:\Descargas\john-1.9.0-jumbo-1-win64\run
λ john --test --format=HMAC-SHA512
Will run 16 OpenMP threads
Benchmarking: HMAC-SHA512 [password is key, SHA512 256/256 AVX2 4x]... (16xOMP) DONE
Many salts:      18465K c/s real, 45649K c/s virtual
Only one salt:   10304K c/s real, 22885K c/s virtual

G:\Descargas\john-1.9.0-jumbo-1-win64\run
λ john --test --format=PBKDF2-HMAC-SHA256
Will run 16 OpenMP threads
Benchmarking: PBKDF2-HMAC-SHA256 [PBKDF2-SHA256 256/256 AVX2 8x]... (16xOMP) DONE
Speed for cost 1 (iteration count) of 1000
Raw:      135858 c/s real, 122880 c/s virtual

G:\Descargas\john-1.9.0-jumbo-1-win64\run
λ john --test --format=bcrypt
Will run 16 OpenMP threads
Benchmarking: bcrypt ("12345678", 32 iterations) [Blowfish 32/64 X3]... (16xOMP) DONE
Speed for cost 1 (iteration count) of 32
Raw:      19326 c/s real, 20076 c/s virtual

G:\Descargas\john-1.9.0-jumbo-1-win64\run
λ john --test --format=LM
Will run 16 OpenMP threads
Benchmarking: LM [DES 256/256 AVX2]... (16xOMP) DONE
Raw:      168786K c/s real, 715653K c/s virtual

G:\Descargas\john-1.9.0-jumbo-1-win64\run
λ john --test --format=NT
Benchmarking: NT [MD4 256/256 AVX2 8x3]... DONE
Raw:      133860K c/s real, 140004K c/s virtual
```

4º:rompe las contraseñas del zip, (we will rockyou)

Lo primero es sacar la contraseña del zip para poder abrir los archivos que hay dentro. Pero antes vamos a convertir el zip a txt para que “juanito” puede leerlo.

```
kali@kali: ~/Descargas
Archivo Acciones Editar Vista Ayuda
(kali@kali)-[~/Descargas]
$ zip2john shadow.zip > contraseña.txt
(kali@kali)-[~/Descargas]
$ cat contraseña.txt
shadow.zip/shadow.txt:$zip2$*0*3*0*74a43db763e78dcaa45f183ce5d8bbc7*fe7a*161*
0b81c13d829f018ed8d9a52ebb17c018bddc4f27076b031e4ab98ca56a7a94c0930c6a07bae16
1084f347297852cd71dbe7ae3ece93b0bd50c3ca02be8e15f7873f7d8dd8105743b9698ac2774
65a67dc50447403ad6f0b2f36b4fd5e56fc72529e05dadebfc309803cb393cb0ff1453ace95a
ae4cbfcdb7fd37b6f32692e7353a6dee4266a87b167d2a49ad8dcb54389ac5485e0ed479111ff
5859f9a1b5653b1c9df73f44d0f1db9954e2e981bbba8a3a0153a41277a252975b42c6f41c171
cda582d6259b6fb5d64f20fa7c47cce09961789bca87ff3812c0ac68cf4ac38fa9aaaea2a01ef
ac69600a34ec810105da02ff53fadafd4210a9c78431a8f1b60f1f5678ef8ba40ebfc34d8856
5289abf8f1ba840795c9d27aaaddead0afa9ef474f69b9465597d854189cf7488bafca2b7224
66f1655c51326294066d526d3e1d12a4bccf29f5e0f24050c918e52f6226e86b1679534e99a12
da368165346ddc554411a9a6f5d1f7f27*$/zip2$:shadow.txt:shadow.zip:shadow.zip
(kali@kali)-[~/Descargas]
$ john contraseña.txt
Using default input encoding: UTF-8
Loaded 1 password hash (ZIP, WinZip [PBKDF2-SHA1 256/256 AVX2 8x])
Cost 1 (HMAC size) is 353 for all loaded hashes
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
123456 (shadow.zip/shadow.txt)
1g 0:00:00:00 DONE 2/3 (2022-10-27 20:32) 1.587g/s 35874p/s 35874c/s 35874C/s
123456..green
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
(kali@kali)-[~/Descargas]
$
```

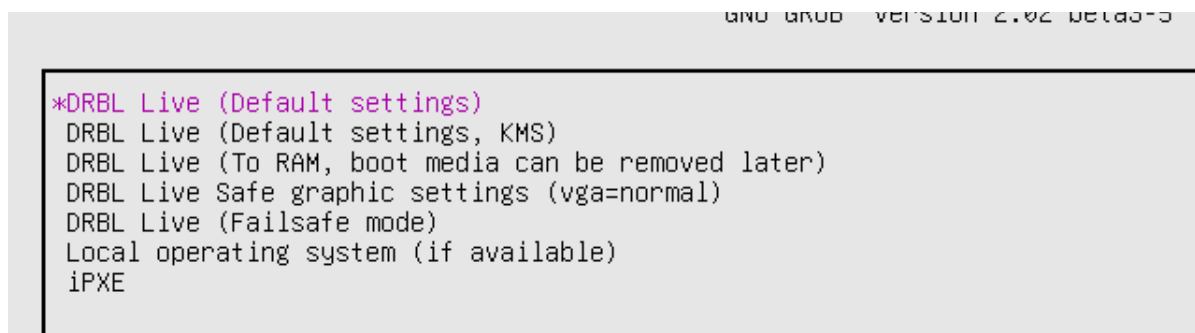
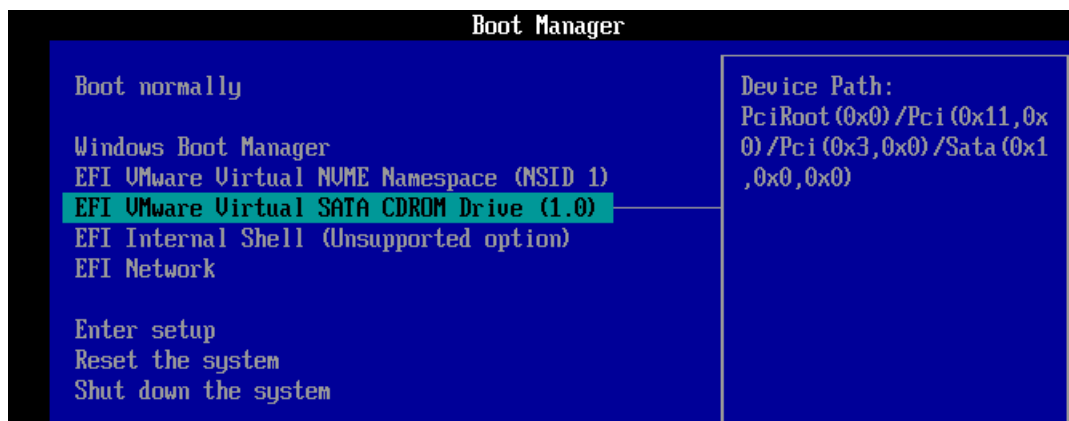
Una vez que ya tenemos la primera contraseña, ya podemos usarla para descomprimir. Para sacar las siguientes cuatro contraseñas usaremos el diccionario rockyou. De forma que no tenga que hacerlo por fuerza bruta, ya que tardaría mucho más.

```
(kali@kali)-[~/Descargas]
$ john shadow.txt -w=rockyou.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with 4 different salts (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Remaining 1 password hash
Cost 1 (iteration count) is 5000 for all loaded hashes
Press 'q' or Ctrl-C to abort, almost any other key for status
administrator (?)
1g 0:00:00:06 DONE (2022-11-02 22:22) 0.1533g/s 2257p/s 2257c/s 2257C/s cassie123..eduardito
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
(kali@kali)-[~/Descargas]
$ john --show shadow.txt
?:whatever
?:dragon
?:princesa
?:administrator

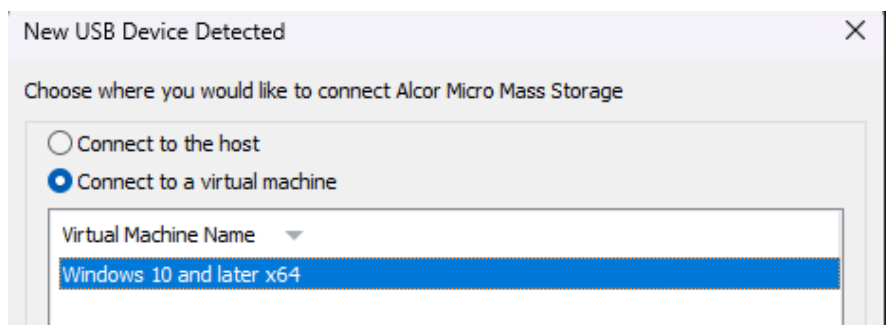
4 password hashes cracked, 0 left
(kali@kali)-[~/Descargas]
$
```

5º: de la maquina de windows sacar los usuarios y contraseñas

Para este ejercicio necesitamos acceder a los ficheros SAM y SYSTEM de la maquina que queremos atacar, para ello deberemos acceder al disco duro donde estas se almacenan. Bien podríamos añadir el disco duro en otra maquina y desde ahí copiar los ficheros. Pero me parecía poco realista, así que he preferido arrancar el ordenador con un live cd y desde ahí copiar a un usb los ficheros que necesitamos.



Conectamos el usb a la maquina virtual.



Y procedemos con los comandos necesarios para copiar los archivos.

```
root@debian:/# mkdir usb
root@debian:/# mkdir windows
root@debian:/# fdisk -l
Disk /dev/nvme0n1: 20 GiB, 21474836480 bytes, 41943040 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: D69505BE-0149-4B26-B6C3-9229AEA08A14

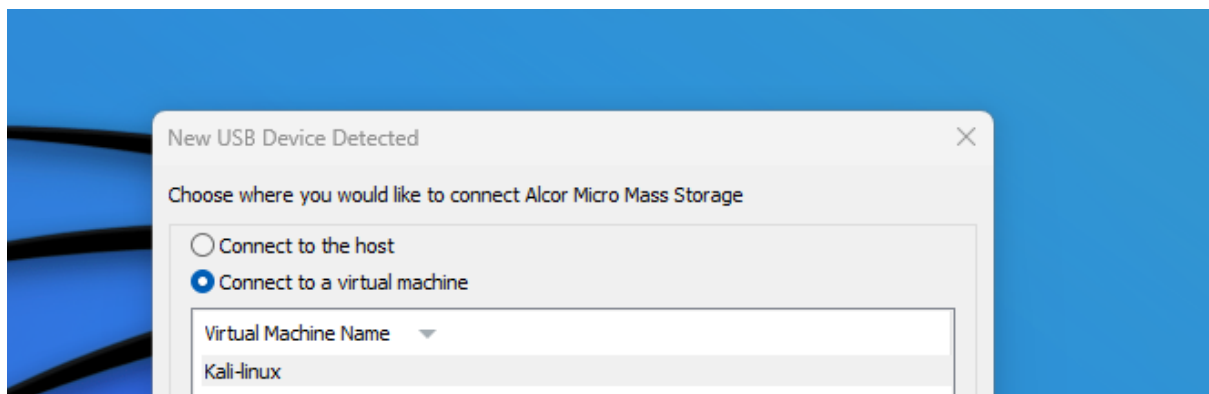
Device            Start      End  Sectors  Size Type
/dev/nvme0n1p1    2048     411647   409600  200M EFI System
/dev/nvme0n1p2   411648     673791   262144  128M Microsoft reserved
/dev/nvme0n1p3   673792  41940991 41267200 19,7G Microsoft basic data

Disk /dev/loop0: 538,6 MiB, 564772864 bytes, 1103072 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/sda: 3,8 GiB, 4026531840 bytes, 7864320 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x1256c2f7

Device    Boot Start      End  Sectors  Size Id Type
/dev/sda1             992 7864319 7863328  3,8G  c W95 FAT32 (LBA)
root@debian:/# mount /dev/sda1 usb
root@debian:/# mount /dev/nvme0n1p3 windows
root@debian:/# cd windows/Windows/System32/config
root@debian:/windows/Windows/System32/config# cat samba
cat: samba: No existe el fichero o el directorio
root@debian:/windows/Windows/System32/config# cp SAM /usb
root@debian:/windows/Windows/System32/config# cp SYSTEM /usb
root@debian:/windows/Windows/System32/config# cd /
root@debian:/# cd usb
root@debian:/usb# ls
SAM  SYSTEM  System Volume Information
root@debian:/usb# umount usb
umount: usb: mountpoint not found
root@debian:/usb# cd ..
root@debian:/# umount usb
root@debian:/# █
```

Ahora vamos a una maqui linux y convertimos los archivos SAM y SYSTEM a txt para que “juanito” una vez mas sea capaz de leerlos.



```
kali@kali: ~/Escritorio
Archivo Acciones Editar Vista Ayuda
zsh: corrupt history file /home/kali/.zsh_history
(kali@kali)~[~]
$ cd Escritorio
(kali@kali)~[/Escritorio]
$ samdump2 SYSTEM SAM > windows.txt
(kali@kali)~[/Escritorio]
$ cat windows.txt
*disabled* Administrador:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 :::
*disabled* Invitado:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 :::
*disabled* :503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 :::
oscar:1000:aad3b435b51404eeaad3b435b51404ee:b32db29f51c06e0b7ae42f8355dab807 :::
:1001:aad3b435b51404eeaad3b435b51404ee:a2212d8b2bb5a18c1f0f9343956b3f23 :::
:1002:aad3b435b51404eeaad3b435b51404ee:9ab721ecba25a83055ea8ba89c7c717b :::
:1003:aad3b435b51404eeaad3b435b51404ee:edba5cbc1ed46b673a7f0a86fdbfca2f :::
:1004:aad3b435b51404eeaad3b435b51404ee:cf74007f626c01b74c94f0e95423ab50 :::
:1005:aad3b435b51404eeaad3b435b51404ee:93609de8c3eeaa8113d33131230a192f :::
:1006:aad3b435b51404eeaad3b435b51404ee:6a5d34a235d3553fb8aa3b1c036ff684 :::
:1007:aad3b435b51404eeaad3b435b51404ee:186cb09181e2c2ecaac768c47c729904 :::
:1008:aad3b435b51404eeaad3b435b51404ee:cadceffac32144a358cbd798d8c132f :::
```

Ahora podríamos seguir el ejercicio en la maquina linux, pero he preferido hacerlo en mi windows ya que asi puedo usar todos los nucleos del procesador y agilizar el proceso. Para ello basta con conectar el usb, con los tres archivos, a windows.

Ahora vamos a crear el diccionario, no tiene mucha complejidad más alla de encontrar todos los nombres y los usuarios de github. El diccionario tiene este formato, una contraseña por linea.

konwashus
Danielocf
samu530
CarlosEscuderoBejerano
AlejandroMartinez163
Raulestepa22
Israngomez
Jaaviii
Ines-Tonato
PatriciaCaP
joseantonio08
APCWalls
isaac-santi
Andrei-DG
Cesgilher
alvarooesteban
LorenzoNPBB
davidcanadillas
cristianotermin99
israelngomez
Isaac
Isaacsantiago
Isaacsantiagocaceres
david
davidcañadillas
davidcañadillasgarcía
patriciacasaspalomeque
patriciacasas
patricia
samueldelcastilloescorial
samuel
samueldelcastillo
carlos
carlosescudero
carlosescuderoheberano
alvaro
alvaroesteban
alvaroestebangonzalo

Con el diccionario ya hecho solo falta crear un nuevo set de reglas en el archivo de configuracion de "juanito".

```
*john.conf: Bloc de notas

Archivo  Editar  Ver

[List.Rules:kekwa]
:
Az"[2][0][2][2-9]"
Az"[2][0][2][2-9]"c
Az"[2][0][2][2-9]"cse3
Az"[2][0][2][2-9]"se3
Az"[2][0][2][2-9]"csi1
Az"[2][0][2][2-9]"si1
Az"[2][0][2][2-9]"cso0
Az"[2][0][2][2-9]"so0
Az"[2][0][2][2-9]"cse3sa@
Az"[2][0][2][2-9]"se3sa@
Az"[2][0][2][2-9]"cse3si1
Az"[2][0][2][2-9]"se3si1
Az"[2][0][2][2-9]"cse3so0
Az"[2][0][2][2-9]"se3so0
Az"[2][0][2][2-9]"cse3sa@
Az"[2][0][2][2-9]"se3sa@
Az"[2][0][2][2-9]"csi1so0
Az"[2][0][2][2-9]"si1so0
Az"[2][0][2][2-9]"csi1sa@
Az"[2][0][2][2-9]"si1sa@
Az"[2][0][2][2-9]"cso0sa@
Az"[2][0][2][2-9]"so0sa@
Az"[2][0][2][2-9]"cse3si1so0
Az"[2][0][2][2-9]"se3si1so0
Az"[2][0][2][2-9]"cse3si1sa@
Az"[2][0][2][2-9]"se3si1sa@
Az"[2][0][2][2-9]"csi1so0sa@
Az"[2][0][2][2-9]"si1so0sa@
Az"[2][0][2][2-9]"cse3si1so0sa@
Az"[2][0][2][2-9]"se3si1so0sa@
```

Ahora usando el diccionario en conjunción con la nueva lista de reglas, y especificando el formato NTLM, podemos proceder a descryptar las contraseñas.

Dos de ellas salen por fuerza bruta así que no necesitaríamos ni el diccionario ni las reglas.


```
G:\Descargas\john-1.9.0-jumbo-1-win64\run
λ john.exe D:\windows.txt --format=NT --wordlist=D:\diccionario.txt --rules=kekw --fork=12
```

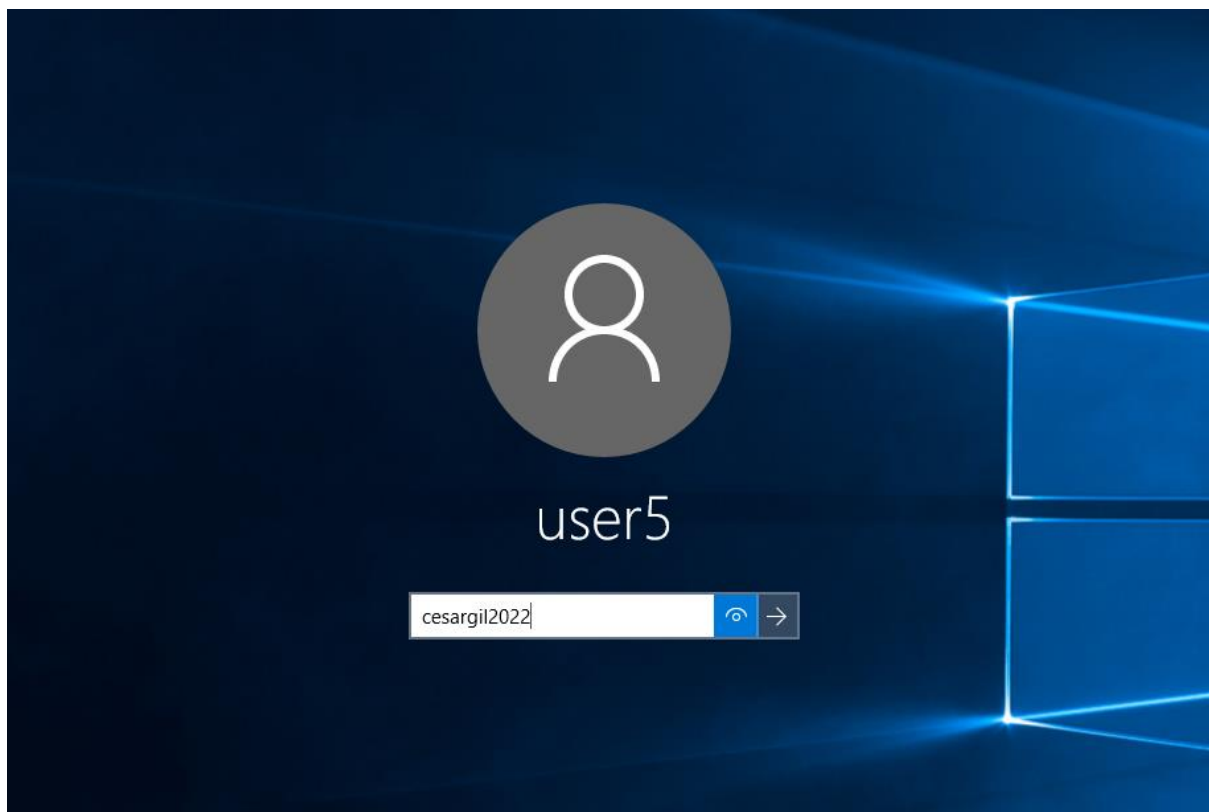
Si comprobamos las contraseñas, vemos que ahí están.

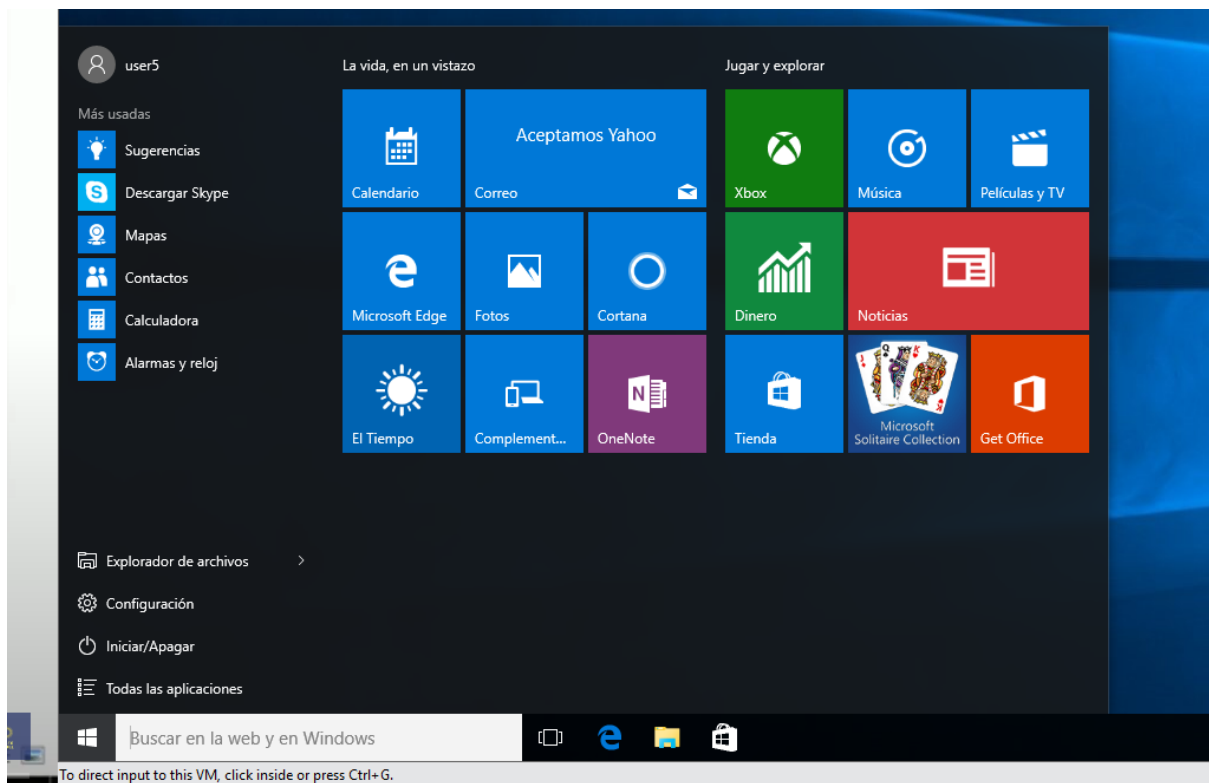
```
G:\Descargas\john-1.9.0-jumbo-1-win64\run
λ john --show --format=NT D:\windows.txt
oscar:oscar:1000:aad3b435b51404eeaad3b435b51404ee:b32db29f51c06e0b7ae42f8355dab807:::
:APCWalls2022:1001:aad3b435b51404eeaad3b435b51404ee:a2212d8b2bb5a18c1f0f9343956b3f23:::
:D@ni3l0cf2022:1002:aad3b435b51404eeaad3b435b51404ee:9ab721ecba25a83055ea8ba89c7c717b:::
:L0r3nz0NPBB2023:1003:aad3b435b51404eeaad3b435b51404ee:edba5cbc1ed46b673a7f0a86fdbfca2f:::
:J@v1erGuerr@2025:1004:aad3b435b51404eeaad3b435b51404ee:cf74007f626c01b74c94f0e95423ab50:::
:cesargil2022:1005:aad3b435b51404eeaad3b435b51404ee:93609de8c3eeaa8113d33131230a192f:::
:carlos2023:1006:aad3b435b51404eeaad3b435b51404ee:6a5d34a235d3553fb8aa3b1c036ff684:::
:a:1007:aad3b435b51404eeaad3b435b51404ee:186cb09181e2c2ecaac768c47c729904:::
:Patr1c1aCaP2022:1008:aad3b435b51404eeaad3b435b51404ee:cadceffac32144a358cbd798d8c132f:::
```

```
9 password hashes cracked, 0 left
```

```
G:\Descargas\john-1.9.0-jumbo-1-win64\run
λ
```

Ahora vamos a conectarnos a la máquina víctima para ver que efectivamente tenemos las contraseñas.





6º:Deshasea las contraseñas del txt dado,
don+dosletrasminúsculas+unamayúscula+dosnumero+
unsimbolo+FIN ej donniK12\$FIN, HELP tomate un
crunch

Lo primero es crear el diccionario con crunch, para ello el comando es el siguiente.
 En windows es algo diferente que en linux, porque para que te reconozca el caracter especial, hay que poner ^^ en vez de simplemente ^.

```
G:\Descargas\crunch
λ crunch_win.exe 12 12 -t don@@,%%^FIN -o crunch.txt
Crunch will now generate the following amount of data: 754010400 bytes
719 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 58000800

crunch: 72% completed generating output
crunch: 100% completed generating output
```

Ahora usamos John de ripper una vez mas pero empleando el nuevo diccionario.

Decir que solo he sido capaz de sacar una de las contraseñas a pesar de haber habilitado el uso de la gpu porque son demasiadas combinaciones y segun la propia maquina hasta el 10-11-2022 no acabaría de probarlas todas.

Formato SHA1

```
C:\cygwin64\run
λ john.exe G:\Descargas\tiposHashes.txt --wordlist=G:\Descargas\crunch\crunch.txt --format=raw-SHA1-openssl
Device 1: NVIDIA GeForce GTX 1070
Using default input encoding: UTF-8
Loaded 1 password hash (raw-SHA1-openssl [SHA1 OpenSSL])
Note: This format may be a lot faster with --mask acceleration (see doc/MASK).
Press 'q' or Ctrl-C to abort, almost any other key for status
donpa099,FIN (?)
1g 0:00:00:01 DONE (2022-11-06 12:01) 0.5128g/s 17207Kp/s 17207Kc/s 17207KC/s donpa099,FIN..donpbC00/FIN
Use the "--show --format=raw-SHA1-openssl" options to display all of the cracked passwords reliably
Session completed
```

Formato bcrypt

```
C:\cygwin64\run
λ john.exe G:\Descargas\tiposHashes.txt --wordlist=G:\Descargas\crunch\crunch.txt --format=bcrypt-openssl
Device 1: NVIDIA GeForce GTX 1070
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (bcrypt-openssl [Blowfish OpenSSL])
Cost 1 (iteration count) is 1024 for all loaded hashes
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:12 0,01% (ETA: 2022-11-08 09:08) 0g/s 0p/s 322.2c/s 322.2C/s donaaA00!FIN..donaaB24$FIN
0g 0:00:11:55 0,20% (ETA: 2022-11-10 10:54) 0g/s 160.3p/s 320.6c/s 320.6C/s donabI75-FIN..donabJ99[FIN
0g 0:00:39:56 0,65% (ETA: 2022-11-10 16:22) 0g/s 155.5p/s 311.0c/s 311.0C/s donaeI95@FIN..donaeK19%FIN
0g 0:00:53:59 0,88% (ETA: 2022-11-10 16:40) 0g/s 155.5p/s 312.2c/s 312.2C/s donafW66?FIN..donafX91!FIN
```