

Informe Laboratorio 3

Sección x

Alexis Lema

e-mail: alexis.lema_g@mail.udp.cl

Mayo de 2024

Índice

1. Descripción de actividades	2
2. Desarrollo (PASO 1)	2
2.1. En qué se destaca la red del informante del resto	2
2.2. Explica matemáticamente porqué se requieren más de 5000 paquetes para obtener la pass	3
2.3. Obtiene la password con ataque por defecto de aircrack-ng	3
2.4. Indica el tiempo que demoró en obtener la password	3
2.5. Descifra el contenido capturado	3
2.6. Describe como obtiene la url de donde descargar el archivo	4
3. Desarrollo (PASO 2)	5
3.1. Script para modificar diccionario original	5
3.2. Cantidad de passwords finales que contiene rockyou_mod.dic	6
4. Desarrollo (Paso 3)	7
4.1. Obtiene contraseña con hashcat con potfile	10
4.2. Nomenclatura del output	11
4.3. Obtiene contraseña con hashcat sin potfile	11
4.4. Nomenclatura del output	13
4.5. Obtiene contraseña con aircrack-ng	13
4.6. Identifica y modifica parámetros solicitados por pycrack	14
4.7. Obtiene contraseña con pycrack	14

1. Descripción de actividades

Su informante quiere entregarle la contraseña de acceso a una red, pero desconfía de todo medio para entregársela (aún no llega al capítulo del curso en donde aprende a comunicar una password sin que nadie más la pueda interceptar). Por lo tanto, le entregará un archivo que contiene un desafío de autenticación, que al analizarlo, usted podrá obtener la contraseña que lo permite resolver. Como nadie puede ver a su informante (es informante y debe mantener el anonimato), él se comunicará con usted a través de la redes inalámbricas y de una forma que solo usted, como experto en informática y telecomunicaciones, logrará esclarecer.

1. Identifique cual es la red inalámbrica que está utilizando su informante para enviarle información. Obtenga la contraseña de esa red utilizando el ataque por defecto de aircrack-ng, indicando el tiempo requerido para esto. Descifre el contenido transmitido sobre ella y descargue de Internet el archivo que su informante le ha comunicado a través de los paquetes que usted ha descifrado.
2. Descargue el diccionario de Rockyou (utilizado ampliamente en el mundo del pentesting). Haga un script que para cada string contenido en el diccionario, reemplace la primera letra por su letra en capital y agregue un cero al final de la password.

Todos los strings que comiencen con número toca eliminarlos del diccionario. Indique la cantidad de contraseñas que contiene el diccionario modificado debe llamarse rockyou_mod.dic A continuación un ejemplo de cómo se modifican las 10 primeras líneas del diccionario original.

3. A partir del archivo que descargó de Internet, obtenga la password asociada a la generación de dicho archivo. Obtenga la llave mediante un ataque por fuerza bruta. Para esto deberá utilizar tres herramientas distintas para lograr obtener la password del archivo: hashcat, aircrack-ng, pycrack. Esta última, permite entender paso a paso de qué forma se calcula la contraseña a partir de los valores contenidos en el handshake, por lo que deberá agregar dichos valores al código para obtener la password a partir de ellos y de rockyou_mod.dic. Antes de ejecutar esta herramienta deberá deshabilitar la función RunTest().

Al calcular la password con hashcat utilice dos técnicas: una donde el resultado se guarda en el potfile y otra donde se deshabilita el potfile. Indique qué información retorna cada una de las 2 técnicas, identificando claramente cada campo.

Recuerde indicar los 4 mayores problemas que se le presentaron y cómo los solucionó.

2. Desarrollo (PASO 1)

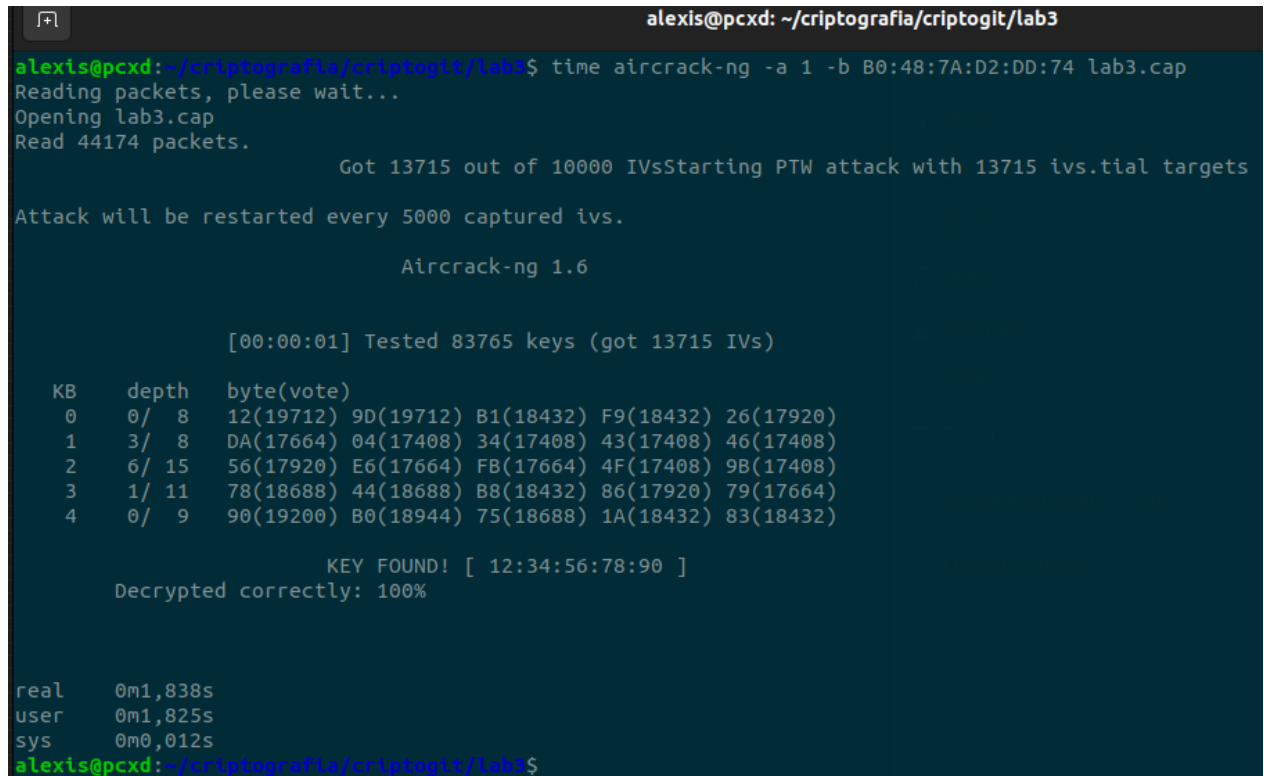
2.1. En qué se destaca la red del informante del resto

Su SSID dicta el protocolo por el cual funciona (es una pista).

2.2. Explica matemáticamente porqué se requieren más de 5000 paquetes para obtener la pass

2.3. Obtiene la password con ataque por defecto de aircrack-ng

```
$ time aircrack-ng -a 1 -b B0:48:7A:D2:DD:74 lab3.cap
```



```
alexis@pcxd: ~/criptografia/criptogit/lab3
alexis@pcxd:~/criptografia/criptogit/lab3$ time aircrack-ng -a 1 -b B0:48:7A:D2:DD:74 lab3.cap
Reading packets, please wait...
Opening lab3.cap
Read 44174 packets.
Got 13715 out of 10000 IVsStarting PTW attack with 13715 ivs.tial targets
Attack will be restarted every 5000 captured ivs.
Aircrack-ng 1.6

[00:00:01] Tested 83765 keys (got 13715 IVs)

KB    depth  byte(vote)
0      0/ 8    12(19712) 9D(19712) B1(18432) F9(18432) 26(17920)
1      3/ 8    DA(17664) 04(17408) 34(17408) 43(17408) 46(17408)
2      6/ 15   56(17920) E6(17664) FB(17664) 4F(17408) 9B(17408)
3      1/ 11   78(18688) 44(18688) B8(18432) 86(17920) 79(17664)
4      0/ 9    90(19200) B0(18944) 75(18688) 1A(18432) 83(18432)

KEY FOUND! [ 12:34:56:78:90 ]
Decrypted correctly: 100%

real    0m1,838s
user    0m1,825s
sys     0m0,012s
alexis@pcxd:~/criptografia/criptogit/lab3$
```

Figura 1: key obtenida

Se obtiene la llave:

```
KEY FOUND! [ 12:34:56:78:90 ]
```

2.4. Indica el tiempo que demoró en obtener la password

Como se aprecia en la figura 1, se tarda 1,838 segundos.

2.5. Descifra el contenido capturado

Se descifra usando el siguiente comando:

```
$ sudo airdecap-ng -w 12:34:56:78:90 lab3.cap
```

2.6 Describe como obtiene la url de donde descargar el archivo DESARROLLO (PASO 1)

```
alexis@pcxd: ~/criptografia/criptogit/lab3
alexis@pcxd:~/criptografia/criptogit/lab3$ time sudo airdecap-ng -w 12:34:56:78:90 lab3.cap
[sudo] password for alexis:
Total number of stations seen          15
Total number of packets read          44174
Total number of WEP data packets      13715
Total number of WPA data packets       60
Number of plaintext data packets       0
Number of decrypted WEP packets       13715
Number of corrupted WEP packets        0
Number of decrypted WPA packets        0
Number of bad TKIP (WPA) packets       0
Number of bad CCMP (WPA) packets       0

real    0m3.307s
user    0m0.060s
sys     0m0.048s
alexis@pcxd:~/criptografia/criptogit/lab3$
```

Figura 2: Descifrado

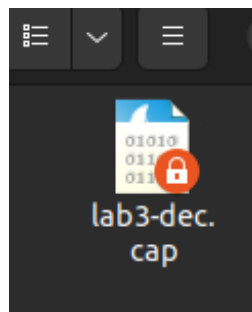


Figura 3: Archivo .cap descifrado

2.6. Describe como obtiene la url de donde descargar el archivo

Luego, revisando el archivo **lab3-dec.cap** a través de la aplicación Wireshark, se obtiene la url *bit.ly/wpa2_* ubicada como texto plano en el payload de los paquetes descifrados.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.11.15	192.168.11.1	ICMP	80	Echo (ping) request id=0x0002, seq=1616/20486, ttl=64 (no response yet)
2	0.026051	192.168.11.15	192.168.11.1	ICMP	80	Echo (ping) request id=0x0002, seq=1617/20742, ttl=64 (no response yet)
3	0.054908	192.168.11.15	192.168.11.1	ICMP	80	Echo (ping) request id=0x0002, seq=1619/21254, ttl=64 (no response yet)
4	0.230540	192.168.11.15	192.168.11.1	ICMP	80	Echo (ping) request id=0x0002, seq=1620/21510, ttl=64 (no response yet)
5	0.241143	192.168.11.15	192.168.11.1	ICMP	80	Echo (ping) request id=0x0002, seq=1621/21766, ttl=64 (no response yet)
6	0.243133	192.168.11.15	192.168.11.1	DNS	101	Standard query 0xf52d A ssl.gstatic.com
7	0.251480	192.168.11.15	192.168.11.1	ICMP	80	Echo (ping) request id=0x0002, seq=1622/22022, ttl=64 (no response yet)
8	0.425344	192.168.11.15	192.168.11.1	ICMP	80	Echo (ping) request id=0x0002, seq=1625/22790, ttl=64 (no response yet)
9	0.514923	192.168.11.1	192.168.11.15	ICMP	80	Echo (ping) reply id=0x0002, seq=1644/27654, ttl=64
10	0.843761	192.168.11.15	192.168.11.1	ICMP	80	Echo (ping) request id=0x0002, seq=1649/28934, ttl=64 (no response yet)
11	0.855329	192.168.11.15	192.168.11.1	ICMP	80	Echo (ping) request id=0x0002, seq=1650/29190, ttl=64 (no response yet)
12	0.857023	192.168.11.15	192.168.11.1	DNS	101	Standard query 0x80db AAAA ssl.gstatic.com
13	0.870072	192.168.11.15	192.168.11.1	ICMP	80	Echo (ping) request id=0x0002, seq=1651/29446, ttl=64 (no response yet)
14	1.037036	192.168.11.15	192.168.11.1	ICMP	80	Echo (ping) request id=0x0002, seq=1652/29702, ttl=64 (no response yet)

Figura 4: Vista wireshark captura descifrada

Colocamos la URL obtenida en el navegador:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	ee:de:67:8c:df:8b	Tp-LinkT_d2:dc:18	802.11	123	Association Request, SN=2292, FN=0, Flags=....., SSID=VTR-16452
2	0.000002	ee:de:67:8c:df:8b	ee:de:67:8c:df:8b (RA)	802.11	10	Acknowledgement, Flags=.....
3	0.002401	Tp-LinkT_d2:dc:18	ee:de:67:8c:df:8b	802.11	102	Association Response, SN=1184, FN=0, Flags=.....
4	0.002402	Tp-LinkT_d2:dc:18	Tp-LinkT_d2:dc:18 (b0:48:7a:d2:dc:18) (RA)	802.11	10	Acknowledgement, Flags=.....
5	0.007381	Tp-LinkT_d2:dc:18	ee:de:67:8c:df:8b	EAPOL	133	Key (Message 1 of 4)
6	0.009336	Tp-LinkT_d2:dc:18	Tp-LinkT_d2:dc:18 (b0:48:7a:d2:dc:18) (RA)	802.11	10	Acknowledgement, Flags=.....
7	0.017080	ee:de:67:8c:df:8b	Tp-LinkT_d2:dc:18	EAPOL	155	Key (Message 2 of 4)
8	0.017082	ee:de:67:8c:df:8b	ee:de:67:8c:df:8b (RA)	802.11	10	Acknowledgement, Flags=.....
9	0.017087	ee:de:67:8c:df:8b	ee:de:67:8c:df:8b (RA)	802.11	10	Clear-to-send, Flags=.....
10	0.050774	Tp-LinkT_d2:dc:18	ee:de:67:8c:df:8b	EAPOL	189	Key (Message 3 of 4)
11	0.050776	Tp-LinkT_d2:dc:18	Tp-LinkT_d2:dc:18 (b0:48:7a:d2:dc:18) (RA)	802.11	10	Acknowledgement, Flags=.....
12	0.054559	ee:de:67:8c:df:8b	Tp-LinkT_d2:dc:18	EAPOL	133	Key (Message 4 of 4)

Figura 5: Captura a descargar

3. Desarrollo (PASO 2)

3.1. Script para modificar diccionario original

Para modificar el diccionario original de Rockyou según lo solicitado, utilizamos un comando en bash que, usando pipes, combina *grep*, *sed*, *tee*, y *wc* para realizar las transformaciones necesarias y contar la cantidad de contraseñas resultantes. El comando completo es:

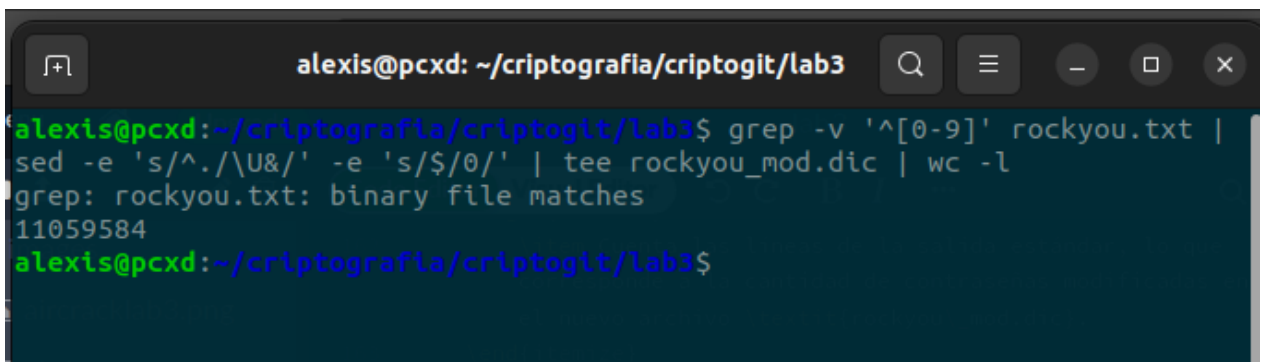
3.2 Cantidad de passwords finales que contiene *rockyou_mod.dic* DESARROLLO (PASO 2)

```
grep -v '^[0-9]' rockyou.txt |  
sed -e 's/^\./\U&/' -e 's/$/0/' |  
tee rockyou_mod.dic |  
wc -l
```

Este comando realiza las siguientes acciones:

- `grep -v '^[0-9]' rockyou.txt`
:
 - Filtra las líneas del archivo *rockyou.txt* eliminando aquellas que comiencen con un número. El operador `-v` indica que se deben excluir las líneas coincidentes.
- `sed -e 's/^\./\U&/' -e 's/$/0/'`
:
 - Aplica dos transformaciones a las líneas filtradas utilizando *sed*.
 - La primera expresión capitaliza la primera letra de cada línea.
 - La segunda expresión `s/$/0/` agrega un cero al final de cada línea.
- `tee rockyou_mod.dic`
:
 - Redirige la salida del comando *sed* tanto al archivo *rockyou_mod.dic* como a la salida estándar, permitiendo su uso en la siguiente parte del pipeline.
- `wc -l`
:
 - Cuenta las líneas de la salida estándar, lo que corresponde a la cantidad de contraseñas modificadas en el nuevo archivo *rockyou_mod.dic*.

3.2. Cantidad de passwords finales que contiene *rockyou_mod.dic*



```
alexis@pcxd: ~/criptografia/criptogit/lab3  
alexis@pcxd:~/criptografia/criptogit/lab3$ grep -v '^[0-9]' rockyou.txt |  
sed -e 's/^\./\U&/' -e 's/$/0/' | tee rockyou_mod.dic | wc -l  
grep: rockyou.txt: binary file matches  
11059584  
alexis@pcxd:~/criptografia/criptogit/lab3$
```

Figura 6: Ejecución comando

El archivo resultante tiene 11059584 passwords.

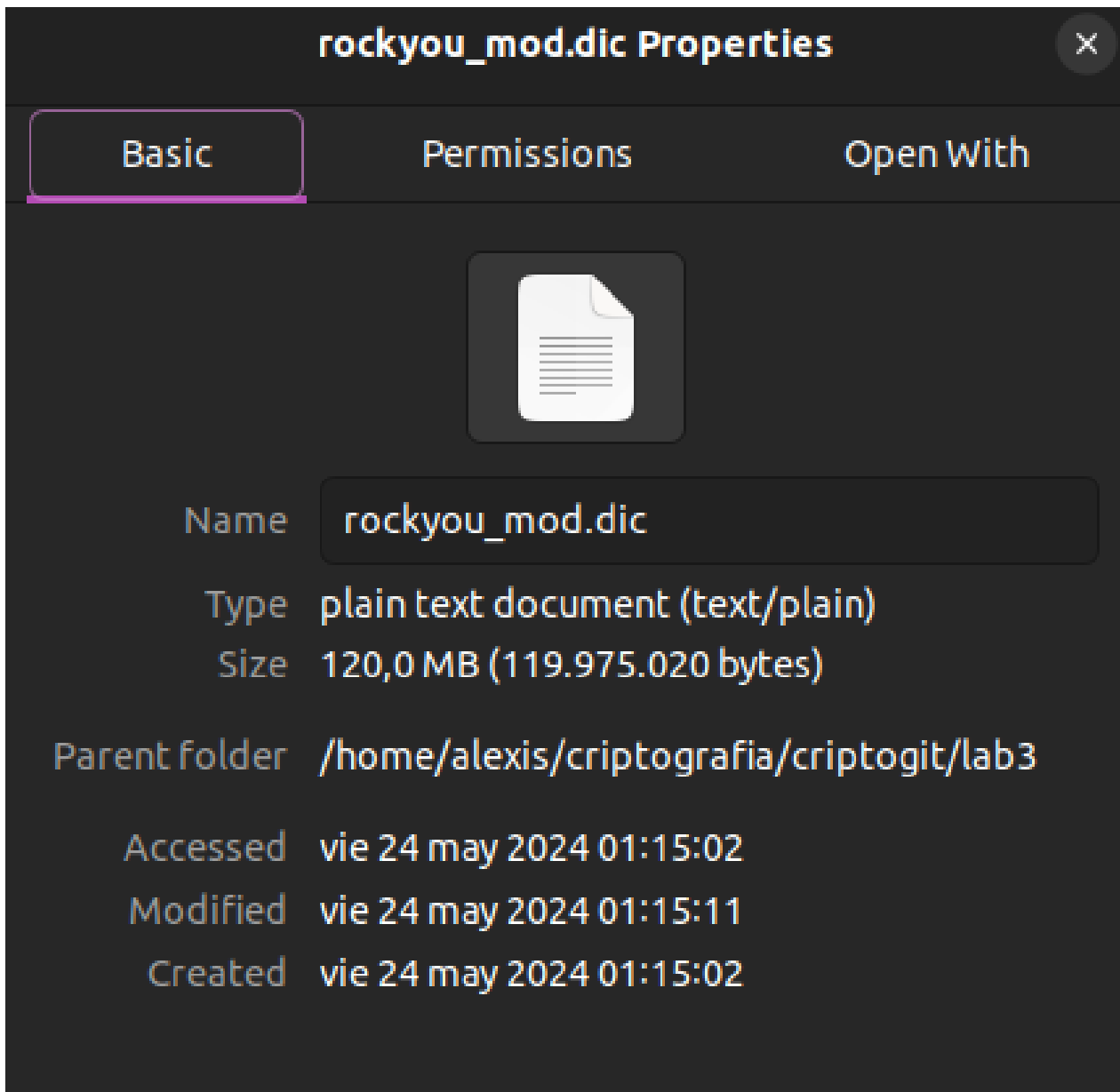


Figura 7: Archivo resultante

4. Desarrollo (Paso 3)

Se obtiene la captura decodificada gracias al convertidor online de hashcap

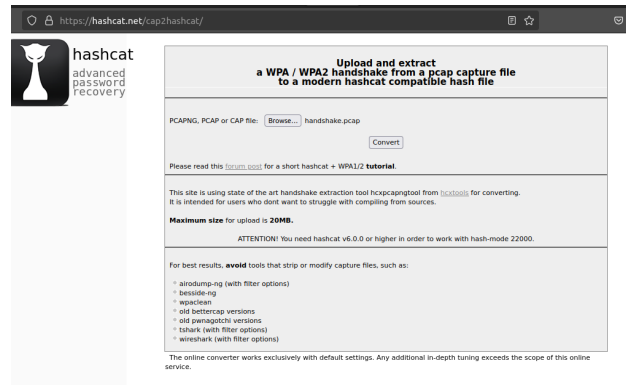


Figura 8: decode online tool

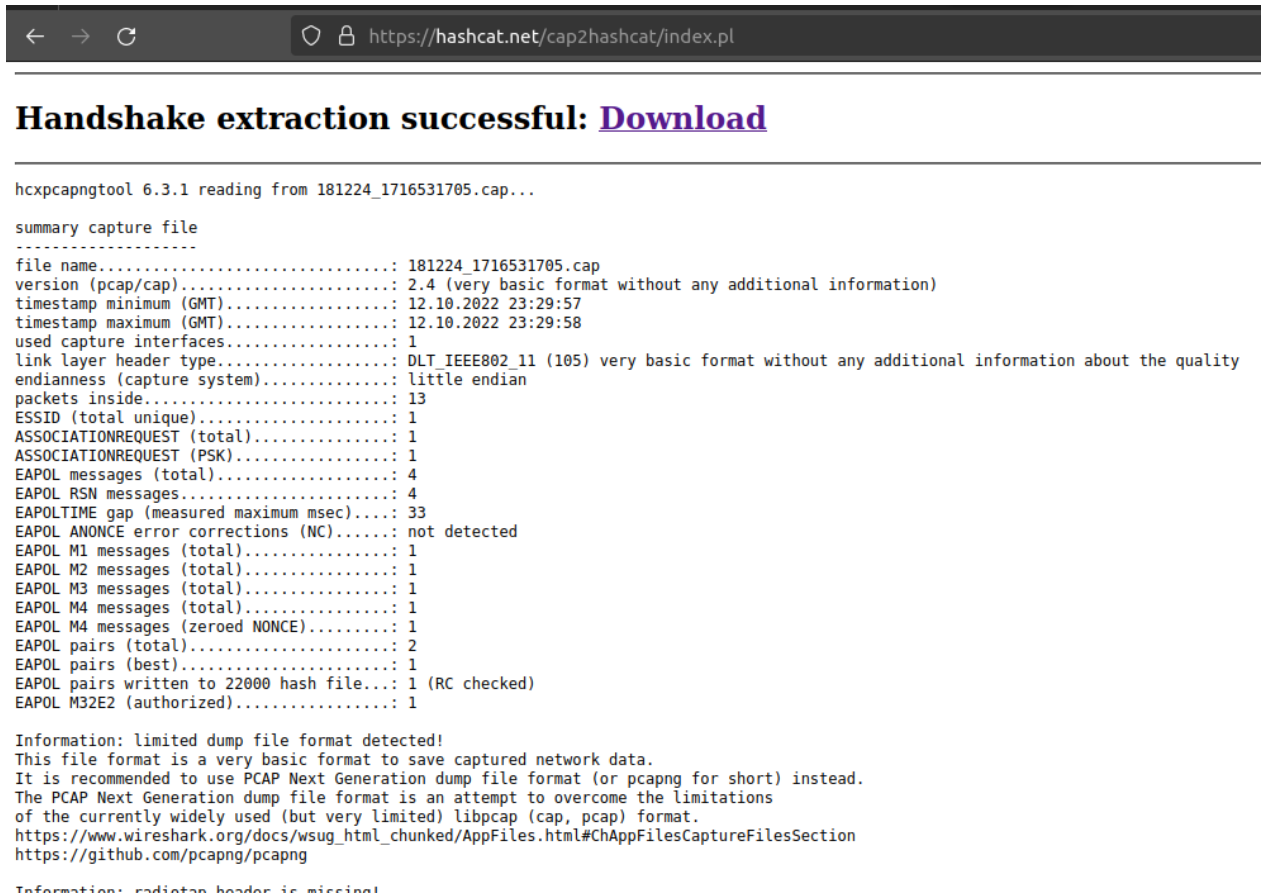


Figura 9: Decode Handshake

Este archivo se renombró como:

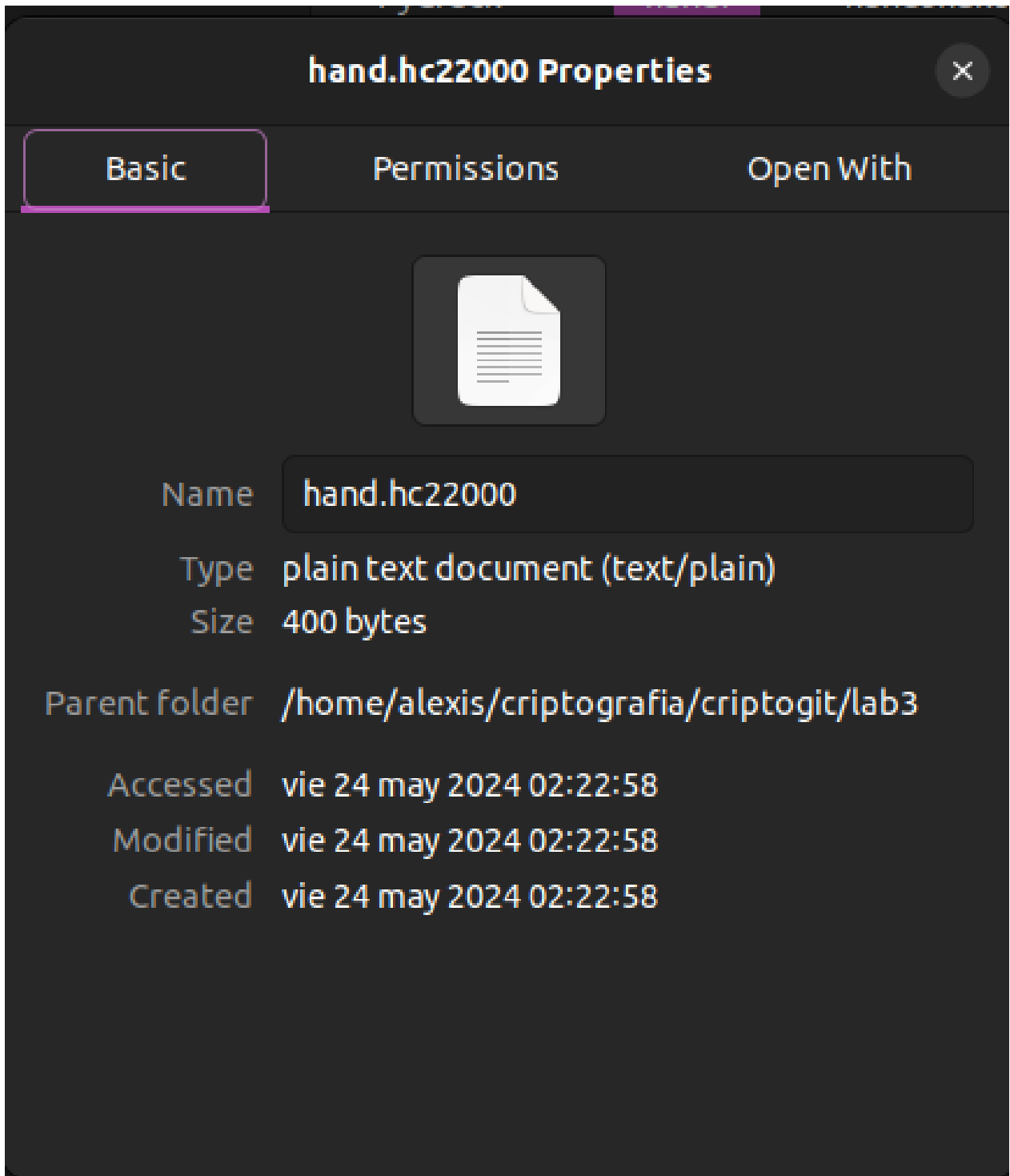


Figura 10: hand file

4.1. Obtiene contraseña con hashcat con potfile

Se tiene el siguiente comando:

```
.\hashcat.exe -m 22000 -a 0 .\hand.hc22000
.\rockyou_mod.dic --potfile-path .\passwords.txt
```

```
PS E:\git\criptolab3\hashcat-6.2.6> .\hashcat.exe -m 22000 -a 0 .\hand.hc22000 .\rockyou_mod.dic --potfile-path .\passwords.txt
hashcat (v6.2.6) starting

Successfully initialized the NVIDIA main driver CUDA runtime library.
Failed to initialize NVIDIA RTC library.

* Device #1: CUDA SDK Toolkit not installed or incorrectly installed.
  CUDA SDK Toolkit required for proper device support and utilization.
  Falling back to OpenCL runtime.

* Device #1: WARNING! Kernel exec timeout is not disabled.
  This may cause "CL_OUT_OF_RESOURCES" or related errors.
  To disable the timeout, see: https://hashcat.net/q/timeoutpatch
OpenCL API (OpenCL 3.0 CUDA 12.4.131) - Platform #1 [NVIDIA Corporation]
=====
* Device #1: NVIDIA GeForce RTX 3060, 12160/12287 MB (3071 MB allocatable), 28MCU

Minimum password length supported by kernel: 8
Maximum password length supported by kernel: 63

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Single-Hash
* Single-Salt
* Slow-Hash-SIMD-LOOP

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 1475 MB

Dictionary cache built:
* Filename...: .\rockyou_mod.dic
* Passwords...: 11059584
* Bytes.....: 119975020
* Keyspace...: 11059577
* Runtime....: 2 secs

1813acb976741b446d43369fb96dbf90:b0487ad2dc18:eede678cdf8b:VTR-1645213:Security0

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 22000 (WPA-PBKDF2-PMKID+EAPOL)
Hash.Target.....: .\hand.hc22000
Time.Started.....: Fri May 24 03:27:36 2024 (0 secs)
Time.Estimated...: Fri May 24 03:27:36 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (.\rockyou_mod.dic)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 382.2 kH/s (9.15ms) @ Accel:128 Loops:128 Thr:32 Vec:1
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 170582/11059577 (1.54%)
Rejected.....: 55894/170582 (32.77%)
Restore.Point....: 0/11059577 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: Password0 -> Ramos120
Hardware.Mon.#1..: Temp: 41c Fan: 0% Util: 95% Core:1957MHz Mem:7300MHz Bus:16

Started: Fri May 24 03:27:16 2024
Stopped: Fri May 24 03:27:37 2024
PS E:\git\criptolab3\hashcat-6.2.6>
```

Figura 11: hashcat con potfile

4.2. Nomenclatura del output

Se logra apreciar: La password hasheada * llave de hash : password texto plano

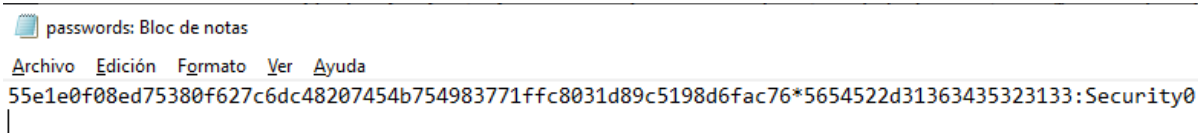


Figura 12: Output potfile

4.3. Obtiene contraseña con hashcat sin potfile

Se modifica el comando anterior, agregando un `-potfile-disable`

```

PS E:\git\criptolab3\hashcat-6.2.6> .\hashcat.exe -m 22000 -a 0 .\hand.hc22000 .\rockyou_mod.dic --potfile-disable
hashcat (v6.2.6) starting

Successfully initialized the NVIDIA main driver CUDA runtime library.
Failed to initialize NVIDIA RTC library.

* Device #1: CUDA SDK Toolkit not installed or incorrectly installed.
  CUDA SDK Toolkit required for proper device support and utilization.
  Falling back to OpenCL runtime.

* Device #1: WARNING! Kernel exec timeout is not disabled.
  This may cause "CL_OUT_OF_RESOURCES" or related errors.
  To disable the timeout, see: https://hashcat.net/q/timeoutpatch
OpenCL API (OpenCL 3.0 CUDA 12.4.131) - Platform #1 [NVIDIA Corporation]
=====
* Device #1: NVIDIA GeForce RTX 3060, 12160/12287 MB (3071 MB allocatable), 28MCU

Minimum password length supported by kernel: 8
Maximum password length supported by kernel: 63

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Single-Hash
* Single-Salt
* Slow-Hash-SIMD-LOOP

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 1475 MB

Dictionary cache hit:
* Filename...: .\rockyou_mod.dic
* Passwords..: 11059577
* Bytes.....: 119975020
* Keyspace...: 11059577

1813acb976741b446d43369fb96dbf90:b0487ad2dc18:eede678cdf8b:VTR-1645213:Security0

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 22000 (WPA-PBKDF2-PMKID+EAPOL)
Hash.Target.....: .\hand.hc22000
Time.Started....: Fri May 24 03:40:33 2024 (0 secs)
Time.Estimated...: Fri May 24 03:40:33 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (.\rockyou_mod.dic)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 376.6 kH/s (9.37ms) @ Accel:4 Loops:512 Thr:256 Vec:1
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 45271/11059577 (0.41%)
Rejected.....: 16599/45271 (36.67%)
Restore.Point....: 0/11059577 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: Password0 -> Dorinel0
Hardware.Mon.#1..: Temp: 39c Fan: 0% Util: 48% Core:1957MHz Mem:7300MHz Bus:16

Started: Fri May 24 03:40:31 2024
Stopped: Fri May 24 03:40:35 2024
PS E:\git\criptolab3\hashcat-6.2.6>

```

Figura 13: hashcat sin potfile

4.4. Nomenclatura del output

Se aprecian las características de la seguridad wifi del cliente, MIC, MAC, el nombre de la red y la password en texto plano

1813acb976741b446d43369fb96dbf90:b0487ad2dc18:eede678cdf8b:VTR-1645213:Security0

4.5. Obtiene contraseña con aircrack-ng

```

alexis@pcxd:~/criptografia/criptogit/lab3$ aircrack-ng -w rockyou_mod.dic handshake.pcap
Reading packets, please wait...
Opening handshake.pcap
Read 13 packets.

# BSSID          ESSID          Encryption
1 B0:48:7A:D2:DC:18 VTR-1645213    WPA (1 handshake)

Choosing first network as target.

Reading packets, please wait...
Opening handshake.pcap
Read 13 packets.

1 potential targets

Aircrack-ng 1.6

[00:00:02] 2736/11059584 keys tested (1226.10 k/s)

Time left: 2 hours, 30 minutes, 17 seconds          0.02%

KEY FOUND! [ Security0 ]

Master Key      : 55 E1 E0 F0 8E D7 53 80 F6 27 C6 DC 48 20 74 54
                  B7 54 98 37 71 FF C8 03 1D 89 C5 19 8D 6F AC 76

Transient Key   : FD FF 61 91 F1 F3 26 71 48 23 D6 DE 05 C0 B2 88
                  DF 64 B2 3C 1B 89 A6 31 30 BA 04 B6 59 D9 7E 65
                  BD D2 07 9E C6 8D 2A D6 EF 7F 9E A1 95 1C BC CC
                  62 A6 5D CC 07 B2 E3 9D 12 99 A7 66 D4 E9 EA 00

EAPOL HMAC     : 18 13 AC B9 76 74 1B 44 6D 43 36 9F B9 6D BF 90

```

Figura 14: Password con aircrack

Password encontrada: Security0

4.6. Identifica y modifica parámetros solicitados por pycrack

Se revisa la información de los paquetes 5, 7, 10 y 12 del archivo wireshark handshake.pcap. Los cuales tienen los campos a modificar que requiere el código **pywd.py** (dictado según los mismos comentarios), se extraen de la siguiente forma:

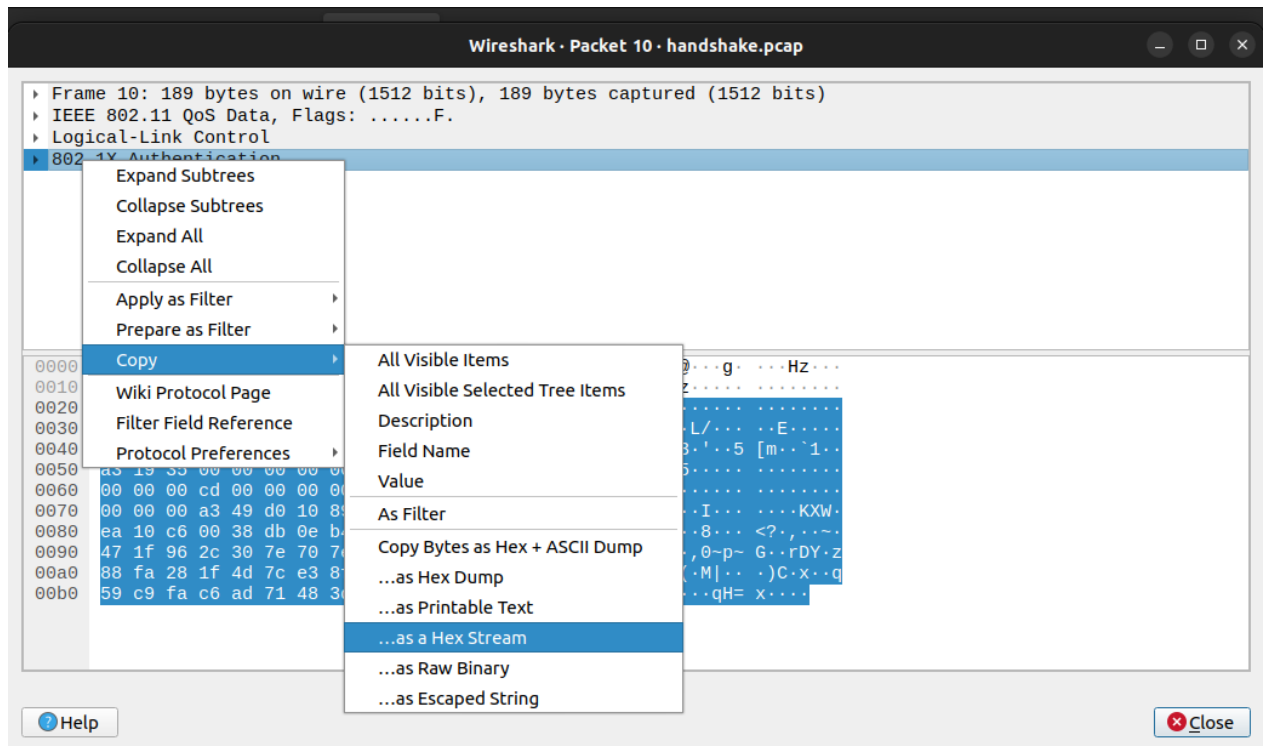


Figura 15: Ejemplo de extracción de información

4.7. Obtiene contraseña con pycrack

```
binascii.Error: Odd-length string
alexis@pcxd:~/criptografia/criptogit/lab3/PyCrack-master$ python3 pywd.py
!!!Password Found!!!
Desired MIC1:      1813acb976741b446d43369fb96dbf90
Computed MIC1:     1813acb976741b446d43369fb96dbf90

Desired MIC2:      a349d01089960aa9f94b5857b0ea10c6
Computed MIC2:     a349d01089960aa9f94b5857b0ea10c6

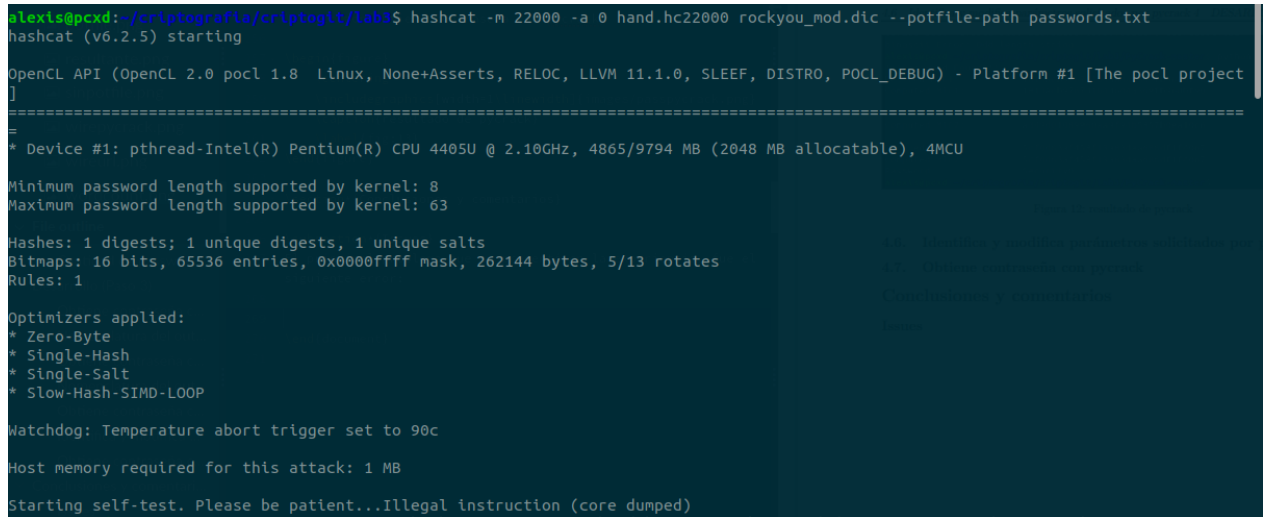
Desired MIC2:      5cf0d63af458f13a83daa686df1f4067
Computed MIC2:     5cf0d63af458f13a83daa686df1f4067
Password:          Security0
alexis@pcxd:~/criptografia/criptogit/lab3/PyCrack-master$
```

Figura 16: resultado de pycrack

Conclusiones y comentarios

Issues

El principal problema que se presentó en la experiencia, fue el siguiente error:



```
alexis@pcxd:~/criptografia/criptogit/lab3$ hashcat -m 22000 -a 0 hand.hc22000 rockyou_mod.dic --potfile-path passwords.txt
hashcat (v6.2.5) starting

OpenCL API (OpenCL 2.0 pocl 1.8 Linux, None+Asserts, RELOC, LLVM 11.1.0, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
=====
+ Device #1: pthread-Intel(R) Pentium(R) CPU 4405U @ 2.10GHz, 4865/9794 MB (2048 MB allocatable), 4MCU

Minimum password length supported by kernel: 8
Maximum password length supported by kernel: 63

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Single-Hash
* Single-Salt
* Slow-Hash-SIMD-LOOP

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 1 MB

Starting self-test. Please be patient...Illegal instruction (core dumped)
```

Figura 17: problema 1: CORE DUMPED

Se presume que fue por la tarjeta grafica integrada del equipo, la cual al ser antigua y no tener los drivers necesarios, no fué ocupada por hashcat y en su defecto, ocupó el procesador del equipo, el cual no satisface la operación en terminos de rendimiento y se terminó matando al proceso hashcat:



Figura 18: Equipo 1 (Notebook Humilde)

Para solucionar esto, se ejecutó el comando de hashcat en un equipo con mayor rendimiento y con grafica dedicada, es por eso que en ciertas partes de la experiencias, hay captura de windows powershell, ya que no hubo otra alternativa

El poderoso equipo en cuestión que me salvó la vida:

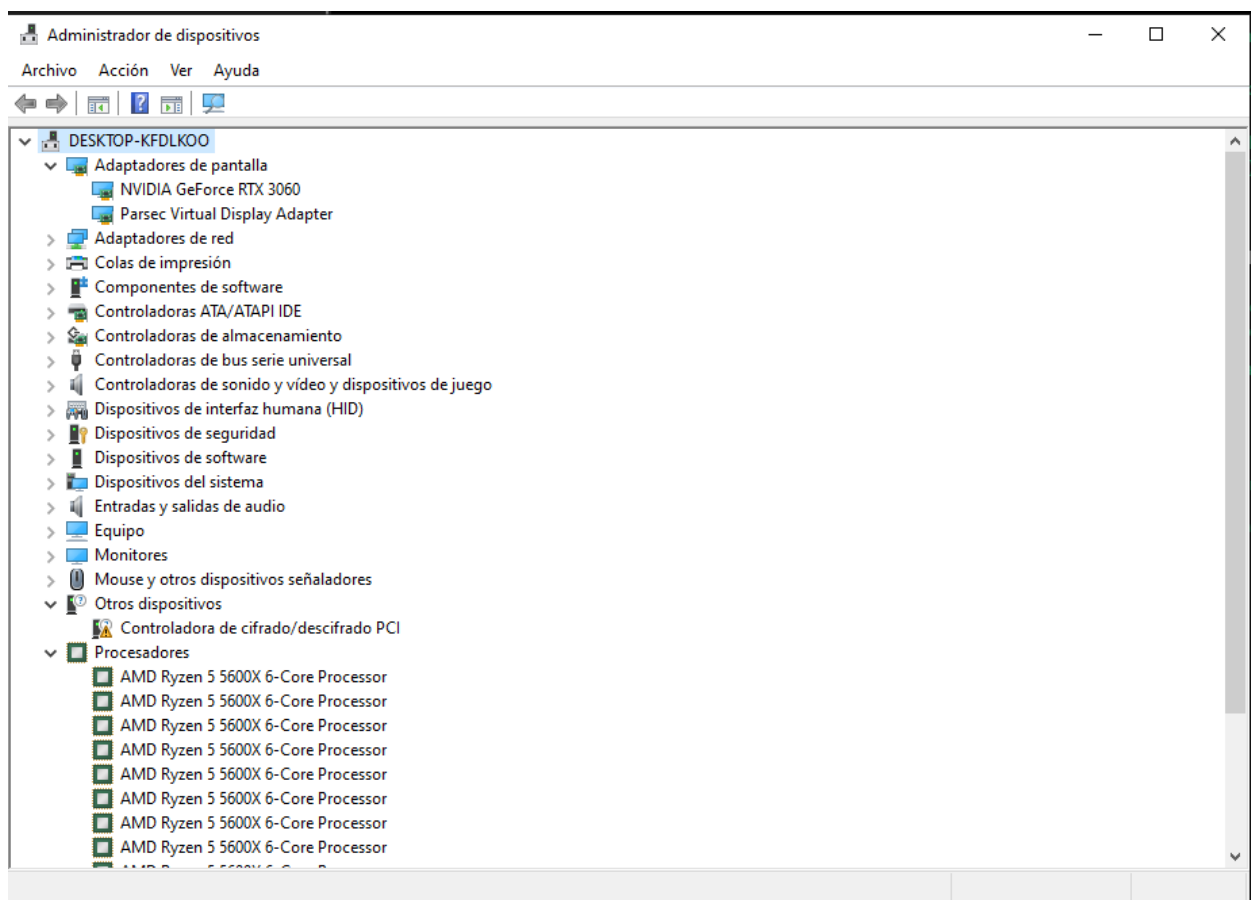


Figura 19: TARRITO

Otro problema fué la captación de paquetes en la primera parte del laboratorio, los cuales no llegaban a los 5000 vectores necesarios