

Informe Laboratorio 3

Sección 1

Pablo Alejandro Díaz Chamorro
e-mail: pablo.diaz_c@mail.udp.cl

Octubre de 2023

Índice

1. Descripción de actividades	2
2. Desarrollo (PASO 1)	2
2.1. identificar en qué se destaca la red del informante del resto	3
2.2. explica matemáticamente porqué se requieren más de 5000 paquetes para obtener la pass	4
2.3. obtiene la password con ataque por defecto de aircrack-ng	5
2.4. indica el tiempo que demoró en obtener la password	6
2.5. descifra el contenido capturado	7
2.6. describe como obtiene la url de donde descargar el archivo	7
3. Desarrollo (PASO 2)	8
3.1. indica script para modificar diccionario original	8
3.2. cantidad de passwords finales que contiene rockyou_mod.dic	9
4. Desarrollo (Paso 3)	9
4.1. obtiene contraseña con hashcat con potfile	9
4.2. identifica nomenclatura del output	10
4.3. obtiene contraseña con hashcat sin potfile	12
4.4. identifica nomenclatura del output	13
4.5. obtiene contraseña con aircrack-ng	15
4.6. identifica y modifica parámetros solicitados por pycrack	16
4.7. obtiene contraseña con pycrack	22

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 RockyouLinks to an external site. (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.
3. Todos los strings que comiencen con número toca eliminarlos del diccionario. Indique la cantidad de contraseñas que contiene el diccionario modificado debe llamarse rock-you_mod.dic A continuación un ejemplo de cómo se modifican las 10 primeras líneas del diccionario original.

2. Desarrollo (PASO 1)

Para comenzar, se procede a cambiar a modo monitor la tarjeta de red WIFI, esto con el objetivo de escuchar el intercambio de información de las redes existentes en el aire.

2.1 identificar en qué se destaca la red del informante del resto DESARROLLO (PASO 1)

```
pablo@felipe:~$ iwconfig
lo                no wireless extensions.
enp4s0            no wireless extensions.
docker0           no wireless extensions.
wlp3s0            IEEE 802.11  ESSID:"Telenatica"
                  Mode:Managed  Frequency:2.437 GHz  Access Point: 98:FC:11:86:B6:B9
                  Bit Rate=6.5 Mb/s   Tx-Power=3 dBm
                  Retry short limit:7   RTS thr:off   Fragment thr:off
                  Power Management:on
                  Link Quality=49/70  Signal level=-61 dBm
                  Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
                  Tx excessive retries:1  Invalid misc:4  Missed beacon:0

pablo@felipe:~$ sudo service NetworkManager stop
pablo@felipe:~$ sudo airmon-ng check

Found 3 processes that could cause trouble.
Kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode

PID Name
605 avahi-daemon
662 wpa_supplicant
671 avahi-daemon

pablo@felipe:~$ sudo airmon-ng check kill

Killing these processes:

PID Name
662 wpa_supplicant
13356 avahi-daemon
13357 avahi-daemon
```

Figura 1: Comandos para cambio de tarjeta de red a modo monitor.

```
pablo@felipe:~$ sudo airmon-ng start wlp3s0

Found 2 processes that could cause trouble.
Kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode

PID Name
13369 avahi-daemon
13370 avahi-daemon

PHY      Interface  Driver      Chipset
phy0     wlp3s0     mt7921e     MEDIATEK Corp. MT7921 802.11ax PCI Express Wireless Network Adapter
          (mac80211 monitor mode vif enabled for [phy0]wlp3s0 on [phy0]wlp3s0mon)
          (mac80211 station mode vif disabled for [phy0]wlp3s0)

pablo@felipe:~$ iwconfig
lo                no wireless extensions.
enp4s0            no wireless extensions.
docker0           no wireless extensions.
wlp3s0mon         IEEE 802.11  Mode:Monitor  Frequency:2.457 GHz  Tx-Power=3 dBm
                  Retry short limit:7   RTS thr:off   Fragment thr:off
                  Power Management:on
```

Figura 2: Comandos para cambio de tarjeta de red a modo monitor.

2.1. identificar en qué se destaca la red del informante del resto

En este caso, primeramente se observan todas las redes existentes en el aire a través del siguiente comando:

```
pablo@pablo-linux:~/Escritorio/s8/criptografia/lab3$ sudo airodump-ng wlp3s0mon
```

Figura 3: Comando para mostrar redes existentes en el aire.

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

2 DESARROLLO (PASO 1)

BSSID	PWR	Beacons	#Data	#/s	CH	MB	ENC	CIPHER	AUTH	ESSID
80:1F:8C:E3:D1:84	-1	0	2	0	1	-1	WPA		<length: 0>	
80:1F:8C:E1:E9:03	-1	0	0	0	0	-1			<length: 0>	
80:1F:8C:E0:E9:93	-1	0	0	0	10	-1			<length: 0>	
C2:8F:EE:0A:50:16	-1	0	0	0	0	-1			<length: 0>	
36:71:AE:1E:F3:95	-50	342	144	0	11	130	WPA2	CCMP	PSK	Grumbly's Wifi
80:4B:7A:D2:D0:74	-50	501	27639	103	0	54e	WEP	WEP	WEP	
52:F8:7B:25:0F:AF	-57	296	4904	0	1	36a	WPA2	CCMP	PSK	AndroidNato
98:FC:11:86:B0:B9	-56	435	285	0	0	130	WPA2	CCMP	PSK	Telematica
58:EF:68:47:59:C6	-62	455	44	0	0	130	WPA2	CCMP	PSK	cableadaTelematica
58:EF:68:47:59:C8	-62	443	13	4	0	130	OPN			cableadaTelematica-Invitado
84:1C:3B:85:E4:07	-77	142	1	0	10	130	WPA2	CCMP	PSK	ZTE_D5EA07
CC:ED:DC:1C:0E:71	-69	136	1	0	13	130	WPA2	CCMP	PSK	JPablo
80:1F:8C:E2:14:A3	-66	303	0	0	11	130	OPN			Alumnos-UDP
80:1F:8C:E2:14:A2	-66	302	0	0	11	130	WPA3	CCMP	OWE	<length: 0>
14:1C:20:E8:E8:35	-71	142	12	0	13	270	WPA2	CCMP	PSK	JPablo_EXT
80:1F:8C:E2:14:A5	-65	302	0	0	11	130	OPN			VIP-UDP
80:1F:8C:E2:14:A6	-66	300	0	0	11	130	WPA3	CCMP	OWE	<length: 0>
80:1F:8C:E2:14:A7	-66	296	0	0	11	130	WPA2	CCMP	MGT	Administrativos-UDP
80:1F:8C:E2:14:A4	-68	300	325	0	11	130	WPA3	CCMP	OWE	_owethn_Alumnos-UDP1993294148
80:1F:8C:E2:14:A1	-66	284	0	0	11	130	OPN			Invitados-UDP
80:1F:8C:E2:14:A0	-66	291	0	0	11	130	WPA3	CCMP	SAE	Sala Hibrida-UDP
8A:DB:1B:C6:83:E9	-71	156	0	0	2	195	WPA2	CCMP	PSK	<length: 0>
CC:D4:A1:D7:81:D0	-73	26	1	0	13	130	WPA2	CCMP	PSK	HUAWEI-B2368-D781DD
AC:FB:CC:1D:60:60	-76	114	5	0	1	130	WPA2	CCMP	PSK	VTR-9492879
9C:9D:7E:22:19:9D	-76	52	0	0	13	130	WPA2	CCMP	PSK	Kata_rep
8E:49:62:EA:35:57	-74	127	0	0	6	65	WPA2	CCMP	PSK	<length: 0>
E4:AB:89:67:33:90	-74	87	0	0	1	130	WPA2	CCMP	PSK	Otakus depa
C0:05:C2:E3:09:41	-78	122	2	0	11	130	WPA2	CCMP	PSK	CAFA
40:D3:43:D7:8C:61	-78	15	0	0	6	130	WPA2	CCMP	PSK	VTR-9108176-24
A6:97:33:A6:2C:21	-79	25	0	0	1	130	WPA2	CCMP	PSK	Otakus depa
CC:ED:DC:98:F1:22	-80	20	0	0	6	130	WPA2	CCMP	PSK	movistar2,4GHZ_9BF122
7C:DB:98:43:F1:0F	-79	12	0	0	1	130	WPA2	CCMP	PSK	Expedientes
94:DB:1B:C6:83:E9	-71	140	0	0	2	192	WPA2	CCMP	PSK	FAMILIAGL_EXT
10:6F:3F:0E:74:A0	-80	45	3	0	1	270	WPA2	CCMP	PSK	LSA
A4:97:33:A6:2C:21	-80	29	0	0	1	130	WPA2	CCMP	PSK	MOVISTAR_2C1F
5C:02:14:B0:AC:06	-80	2	2	0	4	720	WPA2	CCMP	PSK	Xiaomi_3957
C0:05:C2:7B:08:69	-81	4	0	0	1	130	WPA2	CCMP	PSK	Wifi_Miel
00:94:EC:95:3B:A6	-81	12	0	0	10	130	WPA2	CCMP	PSK	HUAWEI-B612-3BA6
80:1F:8C:E0:E8:85	-83	25	0	0	11	130	OPN			VIP-UDP
80:1F:8C:E0:E8:81	-83	63	0	0	11	130	OPN			Invitados-UDP
80:1F:8C:E0:E8:84	-81	27	0	0	11	130	WPA3	CCMP	OWE	<length: 0>
80:1F:8C:E1:B2:83	-81	29	0	0	11	130	OPN			Alumnos-UDP
48:D3:43:50:56:D9	-81	26	0	0	1	130	WPA2	CCMP	PSK	VTR-4173485
80:1F:8C:E0:E8:80	-84	44	0	0	11	130	WPA3	CCMP	SAE	Sala Hibrida-UDP
E0:AB:89:1C:85:38	-83	2	1	0	11	130	WPA2	CCMP	PSK	ELL
3C:04:6A:07:7B:6E	-82	63	0	0	10	270	WPA2	CCMP	PSK	TP-Link_786E

Figura 4: Redes existentes en el aire capturadas por la tarjeta de red.

Al analizar las redes existentes en el aire, se puede inferir que la red del informante se destaca por sobre las otras debido a los siguientes puntos:

- Se observa que la red del informante es la única que contiene un cifrado WEP, ya que normalmente las redes están cifradas con WPA2 o WPA3.
- El tráfico de red asociado a la red WEP es considerablemente mayor comparado con las demás redes.
- La velocidad de la red es sumamente distinta comparado con las demás redes, en donde se tiene que la red WEP tiene una velocidad de 54e.

A partir de estos puntos se puede inferir que la red del informante es la de ESSID=WEP.

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

El ataque de cumpleaños es un tipo de ataque en el que se aprovechan las propiedades matemáticas relacionadas con la teoría de la probabilidad. Este ataque se utiliza para comprometer la comunicación entre dos o más partes de manera efectiva.

Matemáticamente, el enfoque de este ataque es encontrar dos entradas diferentes, llamadas x_1 y x_2 , para las cuales una función llamada f , produce el mismo resultado, es decir, $f(x_1) = f(x_2)$. A esta pareja de entradas se le llama "colisión".

2.3 obtiene la password con ataque por defecto de aircrack-ng DESARROLLO (PASO 1)

El método utilizado para encontrar una colisión se basa en evaluar repetidamente la función f con distintos valores de entrada, los cuales pueden ser seleccionados al azar. Cabe destacar que se sigue evaluando la función hasta que se encuentre el mismo resultado.

Además, se sabe que si una función $f(x)$ produce cualquiera de H salidas diferentes con igual probabilidad, y si H es bastante grande, se espera que obtener un par de argumentos diferentes, x_1 y x_2 , con $f(x_1) = f(x_2)$ después de evaluar la función aproximadamente 1.25 veces la raíz cuadrada de H veces en promedio.

En este caso, para determinar la cantidad de paquetes necesarios para descubrir una contraseña, se utiliza una fórmula que contiene probabilidad de encontrar una colisión. Esta fórmula utiliza el valor H , que representa el conjunto de valores posibles. La fórmula sería la siguiente:

$$n(p; H) = \sqrt{2H \ln \left(\frac{1}{1-p} \right)} \quad (1)$$

- $n(p; H)$: Es la cantidad de paquetes requeridos para descubrir la contraseña.
- p : Es la probabilidad de encontrar una colisión.
- H : Es la cantidad de valores posibles de WEP.

Para calcular la cantidad de paquetes requeridos para descubrir la contraseña, se asume una probabilidad $p = 0.5$, esto indica que se busca un 50 % de probabilidad de éxito. El conjunto de valores posibles H es igual a 2 elevado a la 24 debido a que WEP toma la clave original de 40 bits y la combina con un vector de inicialización (IV) de 24 bits. Entonces hay 2^{24} combinaciones posibles para el IV de 24 bits.

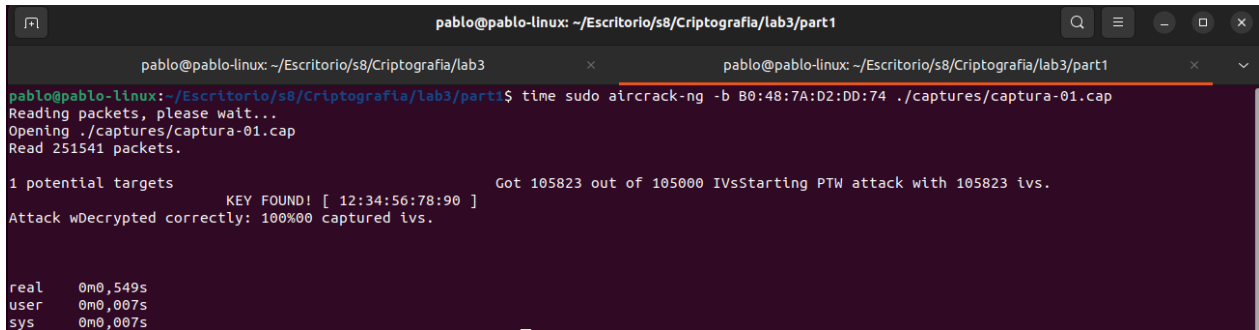
$$n(0,5; 2^{24}) = \sqrt{2 \cdot 2^{24} \cdot \ln \left(\frac{1}{1-0,5} \right)} \quad (2)$$

$$n(0,5; 2^{24}) = 4822,6714 \quad (3)$$

Utilizando la fórmula mencionada anteriormente, se calcula que se necesitan aproximadamente 4822.6714 paquetes. Por último, este valor se redondea a 5000 paquetes capturados para lograr el objetivo de obtener la contraseña, por lo que se requieren más de 5000 paquetes para obtener la pass.

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

Con el fin de obtener la password, se procede a utilizar el ataque por defecto de aircrack-ng, donde se captura y guarda el tráfico de paquetes de la red del informante con la tarjeta de red, utilizando su BSSID=b0:48:7a:d2:dc:18 y el canal 6, a través del siguiente comando:



```
pablo@pablo-linux: ~/Escritorio/s8/Criptografia/lab3/part1
pablo@pablo-linux: ~/Escritorio/s8/Criptografia/lab3
pablo@pablo-linux: ~/Escritorio/s8/Criptografia/lab3/part1$ time sudo aircrack-ng -b B0:48:7A:D2:DD:74 ./captures/captura-01.cap
Reading packets, please wait...
Opening ./captures/captura-01.cap
Read 251541 packets.

1 potential targets
KEY FOUND! [ 12:34:56:78:90 ]
Got 105823 out of 105000 IVsStarting PTW attack with 105823 ivs.
Attack wDecrypted correctly: 100%00 captured ivs.

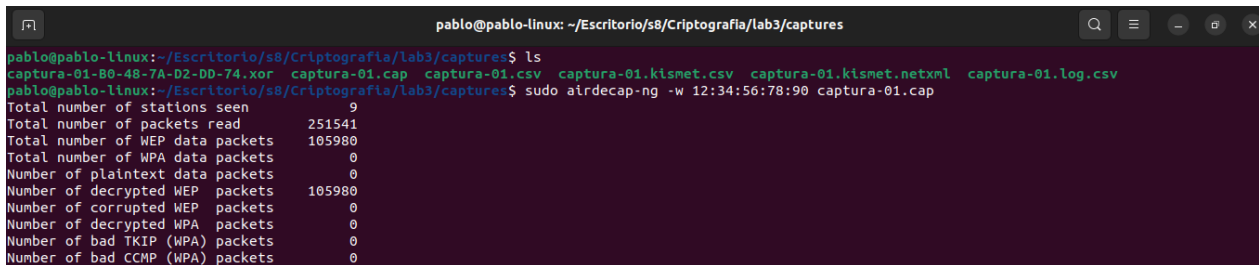
real    0m0,549s
user    0m0,007s
sys     0m0,007s
```

Figura 8: Comando tiempo demora de obtención de password.

En este caso se puede observar que aircrack-ng demoró un tiempo real de 0.549 segundos.

2.5. descifra el contenido capturado

En este caso, para descifrar el contenido que se capturo, se utiliza airdecap-ng, enviando como parámetro la password obtenida anteriormente, para luego ejecutar el siguiente comando:



```
pablo@pablo-linux: ~/Escritorio/s8/Criptografia/lab3/captures
pablo@pablo-linux: ~/Escritorio/s8/Criptografia/lab3/capture$ ls
captura-01-B0-48-7A-D2-DD-74.xor  captura-01.cap  captura-01.csv  captura-01.kismet.csv  captura-01.kismet.netxml  captura-01.log.csv
pablo@pablo-linux: ~/Escritorio/s8/Criptografia/lab3/capture$ sudo airdecap-ng -w 12:34:56:78:90 captura-01.cap
Total number of stations seen      9
Total number of packets read      251541
Total number of WEP data packets  105980
Total number of WPA data packets   0
Number of plaintext data packets   0
Number of decrypted WEP packets    105980
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
```

Figura 9: Comando para descifrar captura de trafico de red del AP.

Después de ejecutar el comando anterior, se guarda el archivo con nombre **captura-01-dec.cap**, el cual es trafico ya descifrado y se puede observar de una forma mas clara.

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

Luego de haber hecho el paso anterior, se procede a abrir el archivo generado anteriormente con Wireshark, esto con el objetivo de analizar el archivo **captura-01-dec.cap** enviado por el informante.

En este caso si seleccionamos un paquete ICMP, se puede observar en la data, la url de donde se debe descargar el archivo **handshake.pcap**, la cual es **bit.ly/wpa2...**

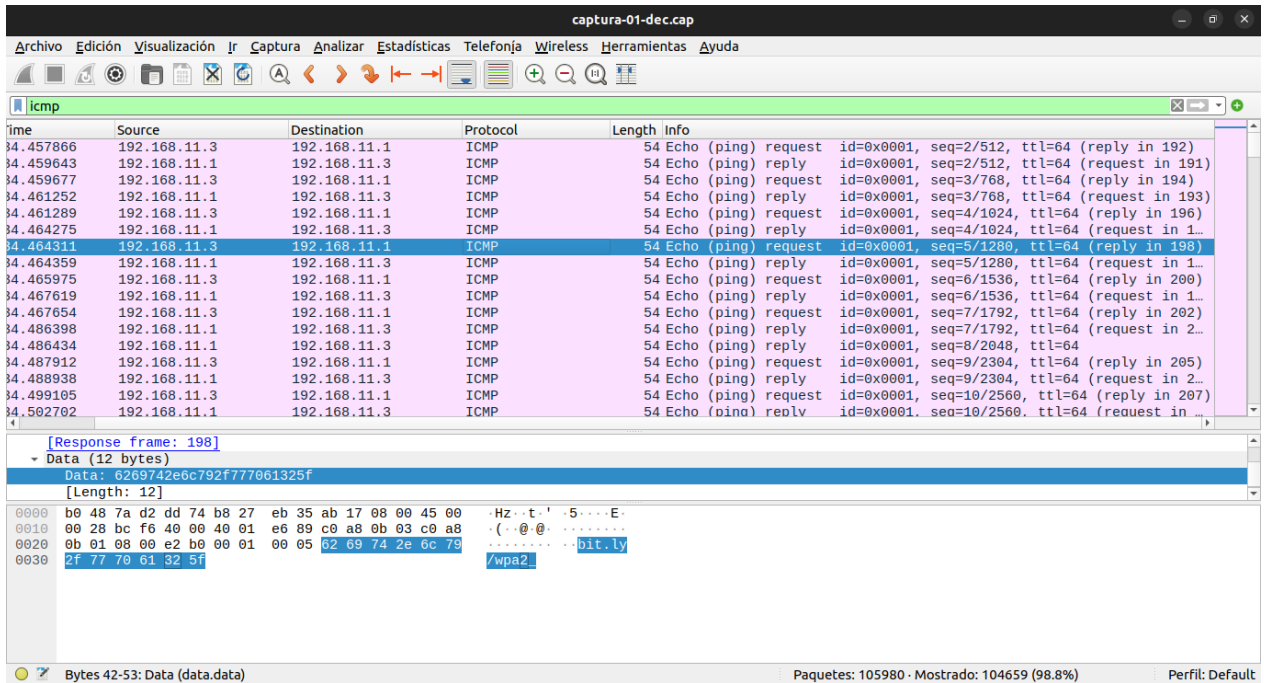


Figura 10: Comando captura de tráfico de red del AP.

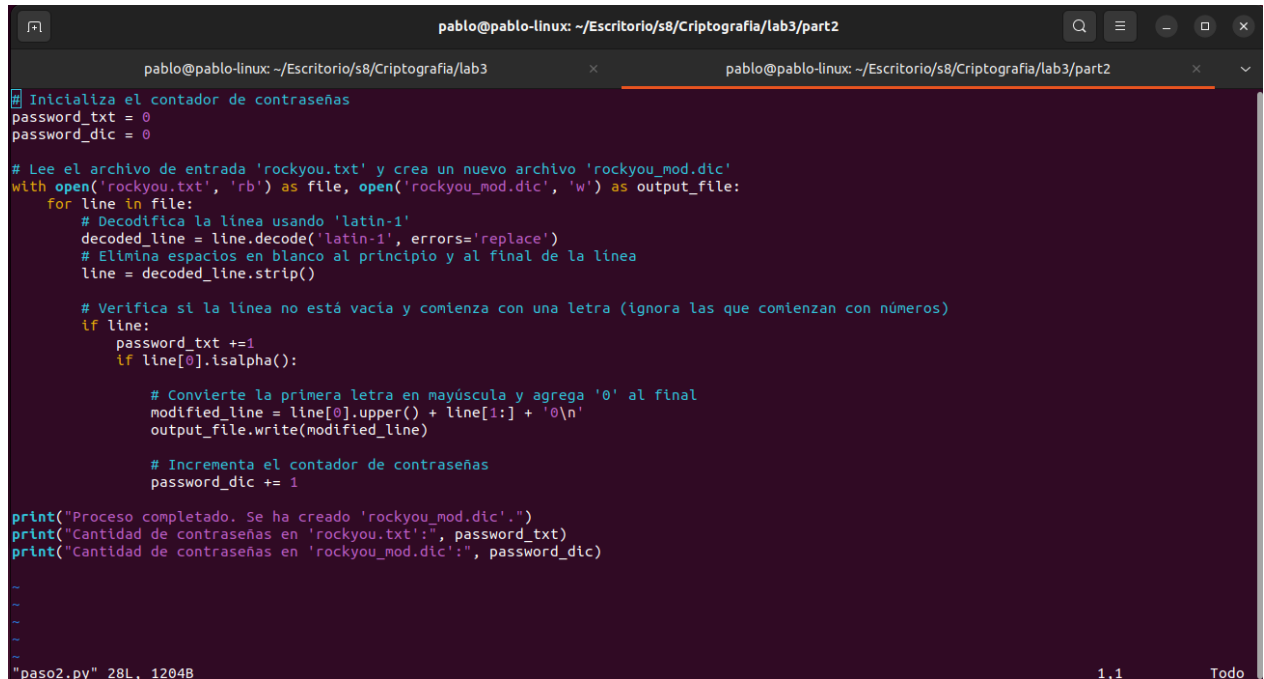
3. Desarrollo (PASO 2)

3.1. indica script para modificar diccionario original

El siguiente script hecho en python, procesa el archivo de entrada **rockyou.txt**, modifica y crea un nuevo archivo llamado **rockyou_mod.dic** según las reglas de reemplazo: La primera letra en mayúscula, eliminando las contraseñas que comiencen con un número y agregando un 0 al final de cada una de las contraseñas.

Por otro lado, también se utiliza un contador para contar cuantas contraseñas se han procesado y cuantas se han escrito en el nuevo archivo.

3.2 cantidad de passwords finales que contiene rockyou_mod.dicDESARROLLO (PASO 3)



```
pablo@pablo-linux: ~/Escritorio/s8/Criptografia/lab3/part2
pablo@pablo-linux: ~/Escritorio/s8/Criptografia/lab3
# Inicializa el contador de contraseñas
password_txt = 0
password_dic = 0

# Lee el archivo de entrada 'rockyou.txt' y crea un nuevo archivo 'rockyou_mod.dic'
with open('rockyou.txt', 'rb') as file, open('rockyou_mod.dic', 'w') as output_file:
    for line in file:
        # Decodifica la línea usando 'latin-1'
        decoded_line = line.decode('latin-1', errors='replace')
        # Elimina espacios en blanco al principio y al final de la línea
        line = decoded_line.strip()

        # Verifica si la línea no está vacía y comienza con una letra (ignora las que comienzan con números)
        if line:
            password_txt += 1
            if line[0].isalpha():
                # Convierte la primera letra en mayúscula y agrega '0' al final
                modified_line = line[0].upper() + line[1:] + '0\n'
                output_file.write(modified_line)

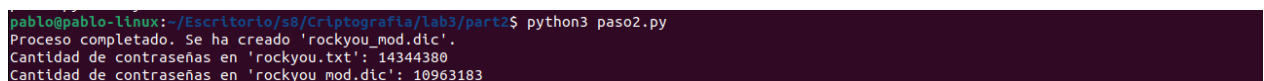
                # Incrementa el contador de contraseñas
                password_dic += 1

print("Proceso completado. Se ha creado 'rockyou_mod.dic'.")
print("Cantidad de contraseñas en 'rockyou.txt':", password_txt)
print("Cantidad de contraseñas en 'rockyou_mod.dic':", password_dic)
```

Figura 11: Script para modificar archivo rockyou.txt.

3.2. cantidad de passwords finales que contiene rockyou_mod.dic

Luego se procede a ejecutar el programa, en el cual como se puede observar que el archivo **rockyou.txt** tiene una cantidad de 14.344.380 contraseñas y se escribieron en el archivo **rockyou_mod.dic** una cantidad de 10.963.183.



```
pablo@pablo-linux: ~/Escritorio/s8/Criptografia/lab3/part2$ python3 paso2.py
Proceso completado. Se ha creado 'rockyou_mod.dic'.
Cantidad de contraseñas en 'rockyou.txt': 14344380
Cantidad de contraseñas en 'rockyou_mod.dic': 10963183
```

Figura 12: Ejecución de script para generación de el archivo **rockyou_mod.dic**.

4. Desarrollo (Paso 3)

4.1. obtiene contraseña con hashcat con potfile

Con el objetivo de obtener la contraseña, se procede a utilizar hashcat para realizar un ataque por fuerza bruta con potfile, esto quiere decir que se utilizara un archivo potfile.txt para registrar contraseñas que se descifran exitosamente. Cabe destacar que este comando utiliza el tipo de ataque, un archivo de hashes y el diccionario modificado **rockyou_mod.dic** de la parte 2. Como se puede inferir el archivo **handshake.pcap** no tiene el formato correcto, por lo que a través de la pagina: <https://hashcat.net/cap2hashcat/>, se procede a convertir el

archivo **handshake.pcap** a un archivo de hashes con formato **.hc22000**. Por esto se procede a utilizar el siguiente comando:

```
pablo@felipe:~/Escritorio$ hashcat -m 22000 198363_1697658097.hc22000 rockyou_mod.dic --potfile-path potfile.txt --force
hashcat (v6.2.5) starting

You have enabled --force to bypass dangerous warnings and errors!
This can hide serious problems and should only be done when debugging.
Do not report hashcat issues encountered when using --force.

Successfully initialized NVIDIA CUDA library.
Failed to initialize NVIDIA RTC library.

* Device #2: CUDA SDK Toolkit not installed or incorrectly installed.
  CUDA SDK Toolkit required for proper device support and utilization.
  Falling back to OpenCL runtime.

nvmlDeviceGetFanSpeed(): Not Supported

OpenCL API (OpenCL 3.0 ) - Platform #1 [Intel(R) Corporation]
=====
* Device #1: Intel(R) UHD Graphics [0x9bc4], 6272/12657 MB (2047 MB allocatable), 23MCU

OpenCL API (OpenCL 3.0 CUDA 12.2.146) - Platform #2 [NVIDIA Corporation]
=====
* Device #2: NVIDIA GeForce GTX 1650, 3776/3903 MB (975 MB allocatable), 14MCU

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
```

Figura 13: Comando de hashcat con potfile para obtener contraseña.

4.2. identifica nomenclatura del output

Después de haber ejecutado el comando anterior, se puede observar la respuesta al ataque de fuerza bruta por hashcat con potfile.

```
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: 2246 MB

Dictionary cache built:
* Filename..: rockyou_mod.dic
* Passwords.: 10963183
* Bytes.....: 118845399
* Keyspace..: 10963176
* Runtime...: 0 secs

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

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 22000 (WPA-PBKDF2-PMKID+EAPOL)
Hash.Target.....: 198363_1697658097.hc22000
Time.Started.....: Wed Oct 18 21:58:55 2023, (1 sec)
Time.Estimated....: Wed Oct 18 21:58:56 2023, (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (rockyou_mod.dic)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 2350 H/s (7.60ms) @ Accel:128 Loops:8 Thr:8 Vec:1
Speed.#2.....: 161.7 kH/s (5.36ms) @ Accel:16 Loops:256 Thr:64 Vec:1
Speed.#*.....: 164.1 kH/s
Recovered.....: 1/1 (100.00%) Digests
Progress.....: 35399/10963176 (0.32%)
Rejected.....: 21063/35399 (59.50%)
Restore.Point....: 0/10963176 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Restore.Sub.#2...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: Elvispresley0 -> Illusions0
Candidates.#2....: Password0 -> Emily060
Hardware.Mon.#1..: N/A
Hardware.Mon.#2..: Temp: 54c Util: 2% Core:1785MHz Mem:6000MHz Bus:16

Started: Wed Oct 18 21:57:50 2023
Stopped: Wed Oct 18 21:58:57 2023
pablo@felipe:~/Escritorio$
```

Figura 14: Resultado ejecución de hashcat con potfile

En este caso, el foco de interés es la respuesta al ataque, por lo que la nomenclatura es la siguiente:

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

Como se observa los parámetros obtenidos están separados por ":", por lo que existen 5 parámetros en la nomenclatura, los cuales son los siguientes:

- 1813acb976741b446d43369fb96dbf90: Este parámetro indica el hash de la contraseña protegida que hashcat intenta descifrar.
- b0487ad2dc18: Este parámetro es el salt, el cual se utiliza para incrementar la seguridad antes del calculo del hash.
- eede678cdf8b: Este parámetro es el PMKID, el cual es un identificador único, el cual se utiliza para identificar y verificar la conexión segura entre el AP y el cliente.
- VTR-1645213: Este parámetro es el nombre de la red a atacar por hashcat.
- Security0: Este parámetro es la contraseña de la red a atacar por hashcat.

4.3. obtiene contraseña con hashcat sin potfile

En esta parte, al igual que en la parte anterior se procede a utilizar hashcat para obtener la contraseña, utilizando los mismos parámetro. No obstante, el ataque por fuerza bruta se realiza sin potfile, esto significa que hashcat no mantendrá un registro de las contraseñas descifradas exitosamente durante la ejecución, por lo que no se crea un archivo **potfile.txt**. Así, se utiliza el siguiente comando:

```
pablo@felipe:~/Escritorio$ hashcat -m 22000 198363_1697658097.hc22000 rockyou_mod.dic --potfile-disable --force
hashcat (v6.2.5) starting

You have enabled --force to bypass dangerous warnings and errors!
This can hide serious problems and should only be done when debugging.
Do not report hashcat issues encountered when using --force.

Successfully initialized NVIDIA CUDA library.
Failed to initialize NVIDIA RTC library.

* Device #2: CUDA SDK Toolkit not installed or incorrectly installed.
  CUDA SDK Toolkit required for proper device support and utilization.
  Falling back to OpenCL runtime.

nvmlDeviceGetFanSpeed(): Not Supported

OpenCL API (OpenCL 3.0 ) - Platform #1 [Intel(R) Corporation]
=====
* Device #1: Intel(R) UHD Graphics [0x9bc4], 6272/12657 MB (2047 MB allocatable), 23MCU

OpenCL API (OpenCL 3.0 CUDA 12.2.146) - Platform #2 [NVIDIA Corporation]
=====
* Device #2: NVIDIA GeForce GTX 1650, 3776/3903 MB (975 MB allocatable), 14MCU

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
```

Figura 15: Comando de hashcat sin potfile para obtener contraseña.

4.4. identifica nomenclatura del output

Al igual que en el caso del ataque de hashcat con potfile, se procede a analizar la nomenclatura de respuesta al ataque, la cual es la siguiente:

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


```
Host memory required for this attack: 2246 MB

Dictionary cache hit:
* Filename..: rockyou_mod.dic
* Passwords.: 10963176
* Bytes.....: 118845399
* Keyspace..: 10963176

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

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 22000 (WPA-PBKDF2-PMKID+EAPOL)
Hash.Target.....: 198363_1697658097.hc22000
Time.Started.....: Wed Oct 18 22:01:06 2023, (2 secs)
Time.Estimated...: Wed Oct 18 22:01:08 2023, (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (rockyou_mod.dic)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 6333 H/s (6.66ms) @ Accel:128 Loops:8 Thr:8 Vec:1
Speed.#2.....: 176.2 kH/s (10.00ms) @ Accel:64 Loops:32 Thr:256 Vec:1
Speed.#*.....: 182.5 kH/s
Recovered.....: 1/1 (100.00%) Digests
Progress.....: 340757/10963176 (3.11%)
Rejected.....: 111381/340757 (32.69%)
Restore.Point....: 0/10963176 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Restore.Sub.#2...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: Laura3330 -> ATHIRAH0
Candidates.#2....: Password0 -> Laura340
Hardware.Mon.#1...: N/A
Hardware.Mon.#2...: Temp: 53c Util: 76% Core:1770MHz Mem:6000MHz Bus:16

Started: Wed Oct 18 22:00:54 2023
Stopped: Wed Oct 18 22:01:09 2023
pablo@felipe:~/Escritorio$
```

Figura 16: Resultado ejecución de hashcat sin potfile

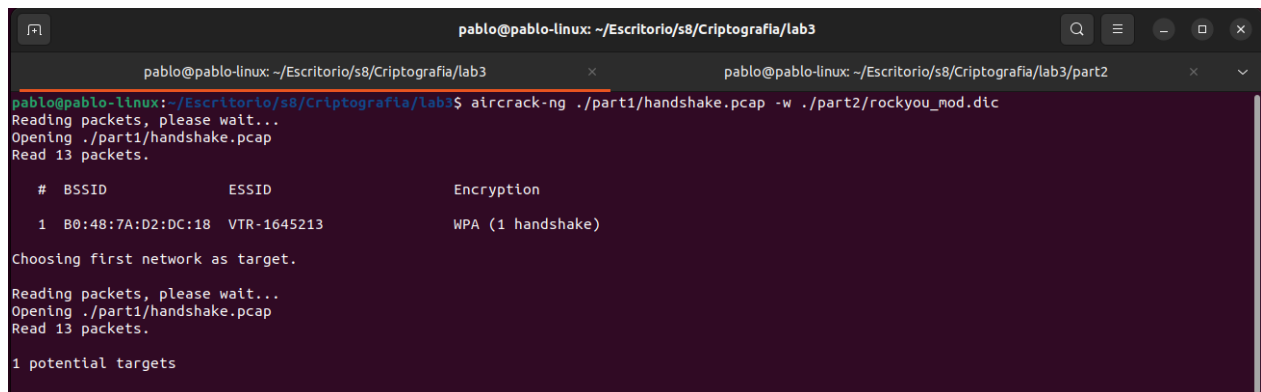
En este caso, se puede observar que la nomenclatura a respuesta del ataque sin potfile, es la misma que a la con potfile, por lo que solo se mencionan los parámetros, los cuales son los siguientes:

- 1813acb976741b446d43369fb96dbf90: Hash.
- b0487ad2dc18: Salt.
- eede678cdf8b: PMKID.
- VTR-1645213: Nombre red.

- Security0: Contraseña red.

4.5. obtiene contraseña con aircrack-ng

En este caso, se utiliza aircrack-ng para realizar el ataque por fuerza bruta, el cual se realiza a través del siguiente comando:



```
pablo@pablo-linux: ~/Escritorio/s8/Criptografia/lab3
pablo@pablo-linux: ~/Escritorio/s8/Criptografia/lab3
pablo@pablo-linux:~/Escritorio/s8/Criptografia/lab3$ aircrack-ng ./part1/handshake.pcap -w ./part2/rockyou_mod.dic
Reading packets, please wait...
Opening ./part1/handshake.pcap
Read 13 packets.

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

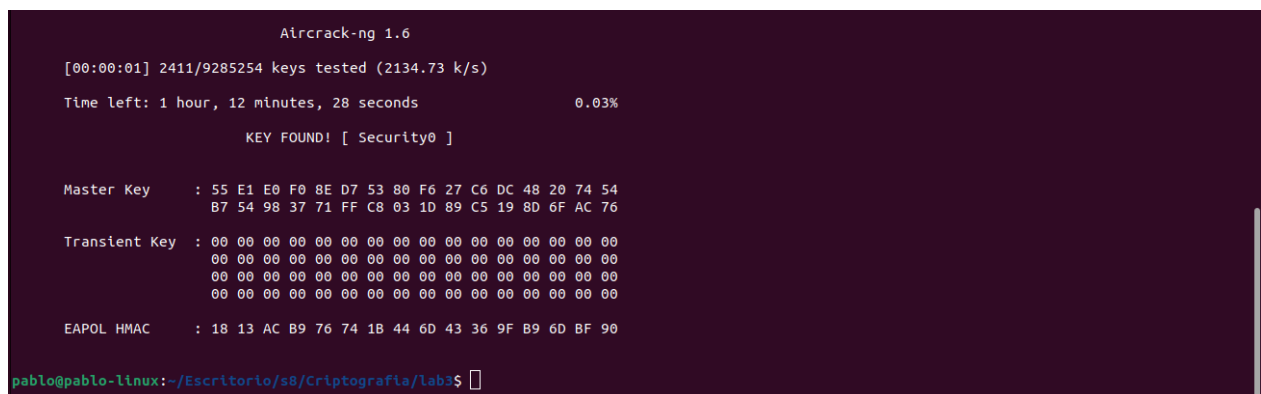
Choosing first network as target.

Reading packets, please wait...
Opening ./part1/handshake.pcap
Read 13 packets.

1 potential targets
```

Figura 17: Comando aircrack-ng para obtener contraseña.

Cabe mencionar que el comando a utilizar, contiene como parámetro el archivo **handshake.pcap** y el diccionario modificado **rockyou_mod.dic** en la parte 2.



```
Aircrack-ng 1.6

[00:00:01] 2411/9285254 keys tested (2134.73 k/s)

Time left: 1 hour, 12 minutes, 28 seconds          0.03%

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 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
              00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
              00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
              00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

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

pablo@pablo-linux:~/Escritorio/s8/Criptografia/lab3$
```

Figura 18: Resultado aircrack-ng obtención contraseña.

Luego de haber ejecutado el comando anterior, se puede observar en la figura anterior que aircrack-ng, entrega como respuesta información relevante:

- [00 : 00 : 01]: Esta parte indica el tiempo transcurrido durante el ataque, por lo que han pasado 1 segundo desde el inicio del ataque.
- 2411/9285254 keys tested (2134.73 k/s): Esto indica información sobre el progreso del ataque, en el que se han probado 2411 claves de un total de 9,285,254 claves posibles.

4.6 *identifica y modifica parámetros solicitados por pycrack 4 DESARROLLO (PASO 3)*

Por otro lado, la velocidad promedio de prueba de claves es de 2,134.73 contraseñas por segundo.

- "Time left: 1 hour, 12 minutes, 28 seconds": Esta parte estima el tiempo para completar el ataque.
- 0.03 %: Esto es el progreso escrito en porcentaje.
- KEY FOUND! [*Security0*]: Esto indica que la clave ha sido encontrada y la clave respectivamente.

Por otro lado, se desprenden tres campos, los cuales son los siguientes:

- Master Key: Este campo indica la clave maestra en hexadecimal, la cual es la que se ha encontrado durante el ataque.
- Transient Key: Esta es una clave temporal utilizada para ejecutar una conexión segura. En este caso, la clave es una serie de ceros, por lo que no se descubrió una clave temporal que sea válida.
- EAPOL HMAC: Este campo indica HMAC(Hash-based Message Authentication Code) asociado a la autenticación EAPOL(Extensible Authentication Protocol over LAN). Por lo que, básicamente es un valor que se utiliza para verificar la integridad de los mensajes de autenticación.

Por lo tanto, como se observa la contraseña obtenida es *Security0* asociado a la red VTR-1645213.

4.6. **identifica y modifica parámetros solicitados por pycrack**

Primeramente, se sabe que pycrack contiene un archivo **pywd.py** de prueba para poder realizar un ataque por fuerza bruta, por lo que se procede a utilizarlo modificando los parámetros respectivos. Cabe mencionar que estos parámetros se obtienen a partir del archivo **handshake.pcap** obtenido en la parte 1.

4.6 identifica y modifica parámetros solicitados por pycrack 4 DESARROLLO (PASO 3)

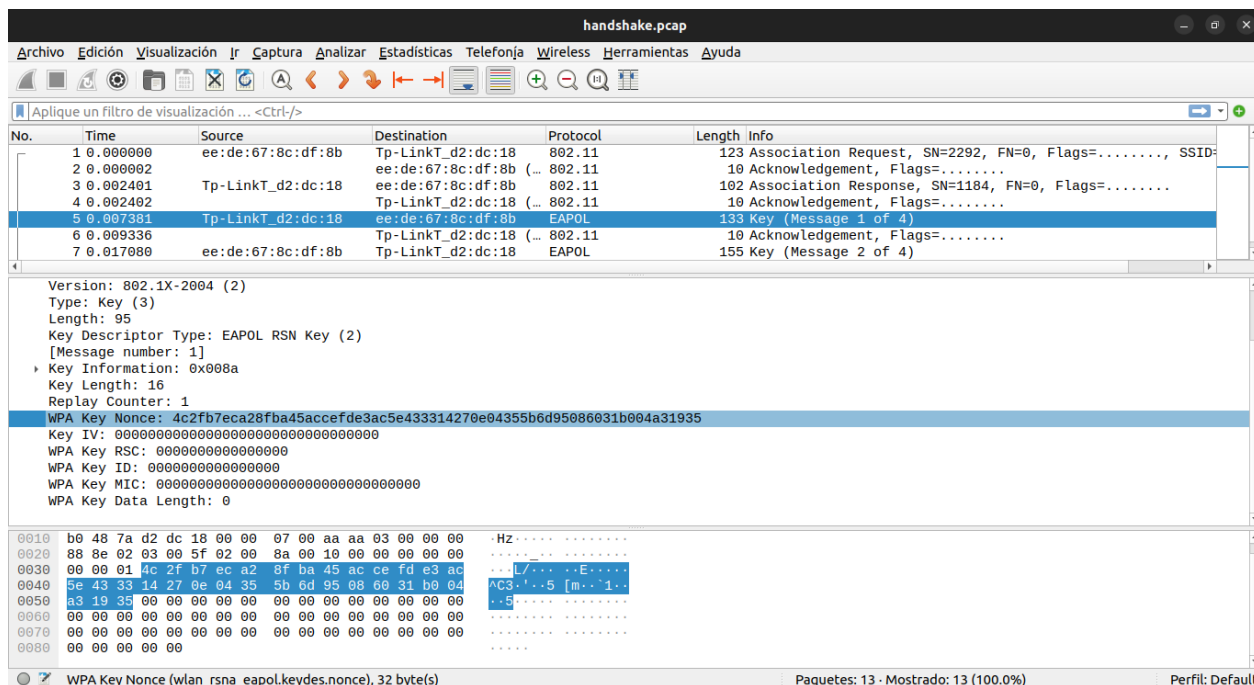


Figura 20: Obtención de parámetro aNonce desde Wireshark.

- El sNonce es también un valor único y aleatorio generado por el cliente. Se utiliza en el intercambio de mensajes junto con el aNonce para establecer una clave temporal (PTK) y garantizar la seguridad de la conexión. Al igual que el caso anterior, se puede obtener a partir del análisis en wireshark del segundo paquete.

4.6 identifica y modifica parámetros solicitados por pycrack 4 DESARROLLO (PASO 3)

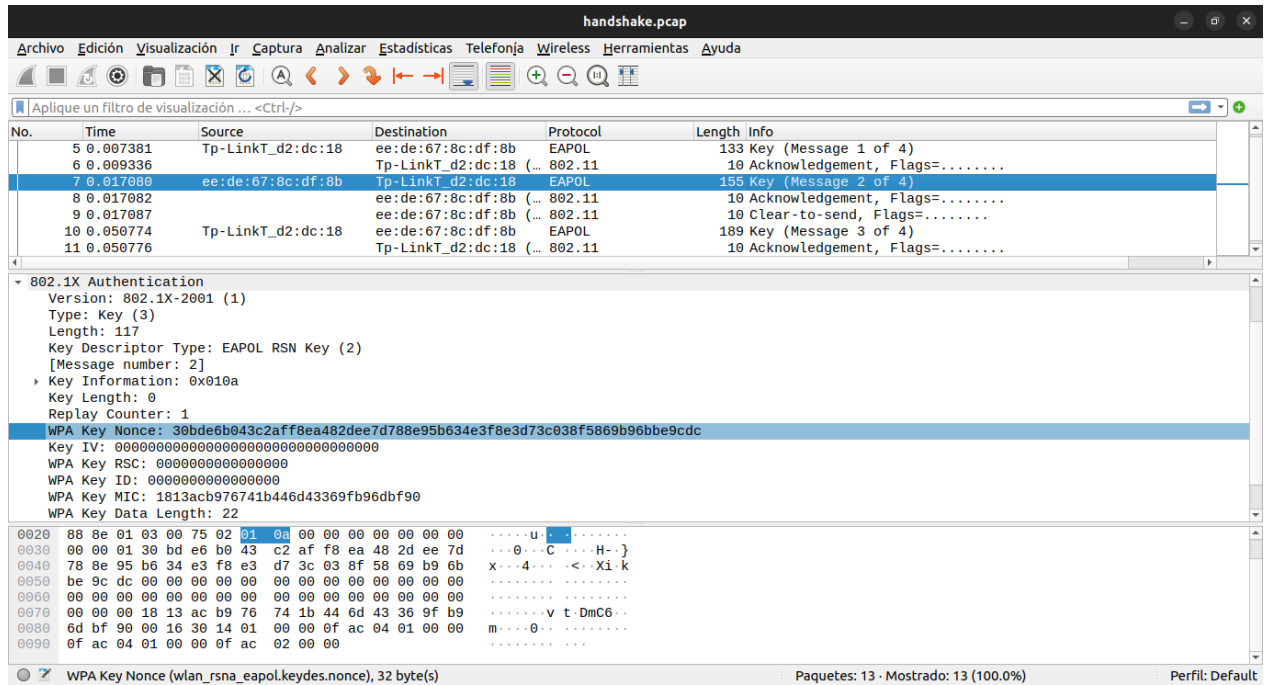


Figura 21: Obtención de parámetro sNounce desde Wireshark.

- Los campos apMAC y cliMAC son básicamente las direcciones MAC del AP y del cliente respectivamente.

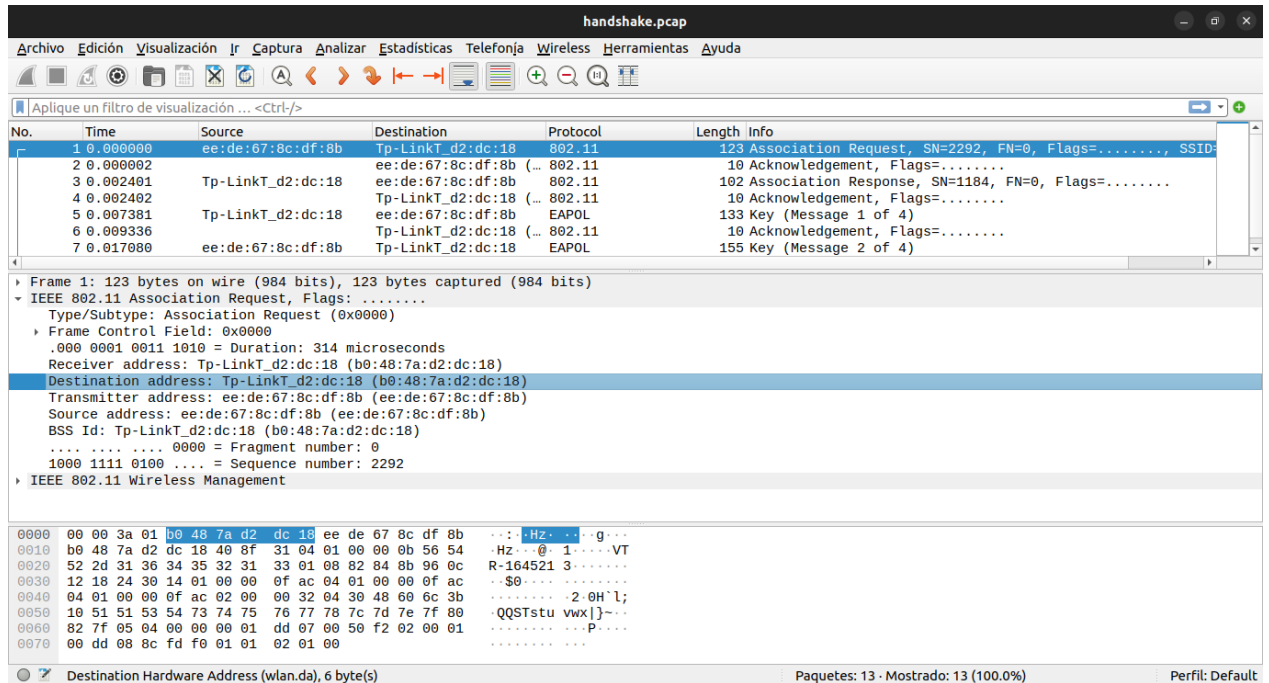


Figura 22: Obtención de parámetro apMAC desde Wireshark.

4.6 identifica y modifica parámetros solicitados por pycrack 4 DESARROLLO (PASO 3)

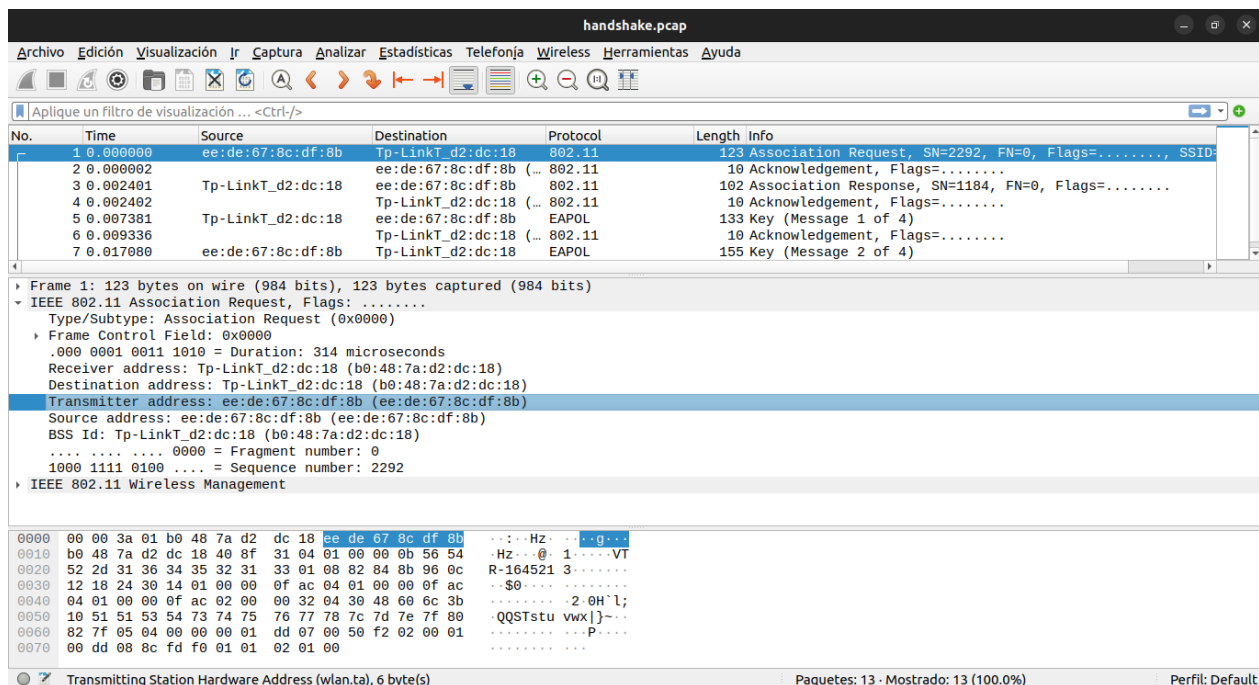


Figura 23: Obtención de parámetro cliMAC desde Wireshark.

- Los campos MIC son valores que se utiliza para verificar la integridad de los mensajes enviados durante el proceso de autenticación. Se calcula a partir de los datos y las claves compartidas entre el cliente y el punto de acceso. Este se agrega a los mensajes para detectar posibles modificaciones no autorizadas mientras se esta transmitiendo o ataques de tipo MiT. En este caso se utilizan los MIC generados en los mensajes 2, 3 y 4 obtenidos desde Wireshark.

4.6 identifica y modifica parámetros solicitados por pycrack 4 DESARROLLO (PASO 3)

Handshaking process

Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda

Aplique un filtro de visualización ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
4	0.002402		Tp-LinkT_d2:dc:18 (...)	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 (...)	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 (...)	802.11	10	Acknowledgement, Flags=.....
9	0.017087		ee:de:67:8c:df:8b (...)	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)

Type: Key (3)
 Length: 117
 Key Descriptor Type: EAPOL RSN Key (2)
 [Message number: 2]
 ▶ Key Information: 0x010a
 Key Length: 0
 Replay Counter: 1
 WPA Key Nonce: 30bde6b043c2aff8ea482dee7788e95b634e3f8e3d73c038f5869b96bbe9cdc
 Key IV: 00000000000000000000000000000000
 WPA Key RSC: 0000000000000000
 WPA Key ID: 0000000000000000
 WPA Key MIC: 1813acb976741b446d43369fb96dbf90
 WPA Key Data Length: 22
 ▶ WPA Key Data: 30140100000fac040100000fac040100000fac020000

```

0020  88 8e 01 03 00 75 02 01 0a 00 00 00 00 00 00 00  ....u.....
0030  00 00 01 30 bd e6 b0 43 c2 af f8 ea 48 2d ee 7d  ...-...C...H-}
0040  78 8e 95 b6 34 e3 f8 e3 d7 3c 03 8f 58 69 b9 6b  x...4...<-Xi-k
0050  be 9c dc 00 00 00 00 00 00 00 00 00 00 00 00  ....
0060  00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
0070  00 00 00 18 13 ac b9 76 74 1b 44 6d 43 36 9f b9  ....v t-DmC6..
0080  6d bf 90 00 16 30 14 01 00 00 0f ac 04 01 00 00  m...-0...
0090  0f ac 04 01 00 00 0f ac 02 00 00  ....
  
```

WPA Key MIC (wlan_rsn_eapol_keydes.mic). 16 byte(s)

Paquetes: 13 · Mostrado: 13 (100.0%)

Perfil: Defa...

Figura 24: Obtención de parámetro MIC desde Wireshark.

- En el caso de los campos datos indican la información intercambiada entre el cliente y el punto de acceso durante el proceso de autenticación. Estos datos contienen los valores aNonce y sNonce, el MIC y otros campos necesarios para establecer una conexión segura. También, al igual que en los campos MIC, se obtiene el campo data a través de la copia del campo 802.1X Authentication como HEX STREAM. Cabe destacar que cada una de las datas se obtiene de los mensajes 2, 3 y 4 respectivamente.

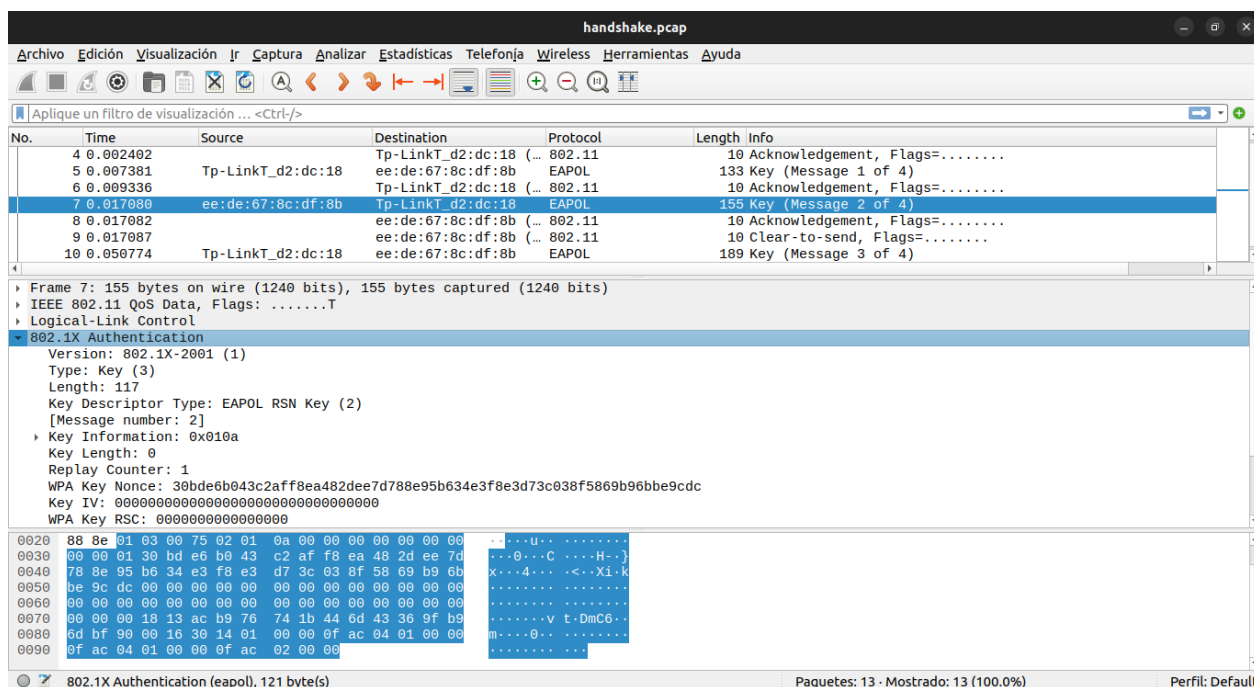


Figura 25: Obtención de parámetro data desde Wireshark.

4.7. obtiene contraseña con pycrack

Luego de haber modificado los parámetros necesarios para realizar el ataque por fuerza bruta, se procede a ejecutar el archivo **pywd.py**.

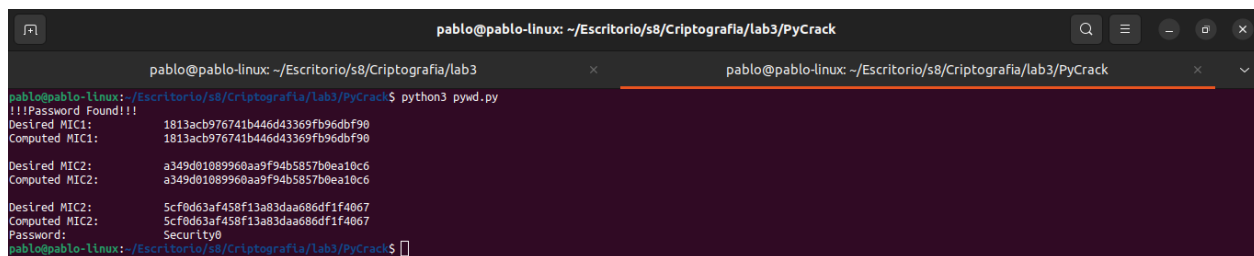


Figura 26: Ejecución de archivo pywd.py.

Como se observa, la password obtenida es Security0.

Conclusiones y comentarios

En esta experiencia de laboratorio, se ha llevado a cabo un proceso completo para obtener la contraseña de una red inalámbrica. A través de una serie de pasos, desde la identificación de la red del informante hasta la obtención de la contraseña, se ha demostrado la importancia

de la seguridad en las redes Wi-Fi y la vulnerabilidad de los sistemas que utilizan cifrado WEP.

El ataque por fuerza bruta se ha utilizado en varias herramientas, como Hashcat, Aircrack-ng y PyCrack, para descifrar la contraseña de la red objetivo. Además, se ha modificado un diccionario de contraseñas comunes para adaptarlo al formato requerido y eliminar contraseñas que comienzan con números.

Además, es de suma importancia de utilizar contraseñas seguras y cifrados fuertes en las redes inalámbricas para evitar este tipo de ataques. La seguridad de las redes Wi-Fi es esencial en un mundo cada vez más digital, donde la privacidad y la integridad de los datos son prioridad. En donde se resalta la importancia de mantener el anonimato y la seguridad al comunicar información sensible a través de redes inalámbricas. Los usuarios deben ser conscientes de las amenazas y saber cómo proteger sus comunicaciones.

Algunos comentarios sobre las herramientas utilizadas son los siguientes:

- La herramienta hashcat tiene una alta capacidad para realizar ataques de fuerza bruta debido a que utiliza la GPU para aumentar la velocidad de ataque y es una herramienta versátil debido a que admite una gran cantidad de algoritmos de cifrado y hash.
- Para aprovechar al máximo la velocidad de Hashcat, es necesario contar con hardware especializado, como tarjetas gráficas de alto rendimiento
- Aircrack-ng es una herramienta más efectiva para cifrados más antiguos, como WEP y WPA. Por el contrario, para cifrados más modernos y seguros, puede ser menos eficiente.
- PyCrack depende de bibliotecas Python, lo que puede implicar en configuraciones adicionales y menor eficiencia en comparación con herramientas mucho mas especializadas.

En conclusión, este informe se ha demostrado distintas vulnerabilidades en las redes inalámbricas, como también la importancia de la seguridad y la protección de datos en un mundo cada vez más conectado. Por otro lado, el uso de las herramientas depende de la necesidad y experiencia del usuario.

GitHub

- <https://github.com/THELUXE1234/laboratorio3cripto>