

Informe Laboratorio 3

Sección 2

Ignacio Guajardo
e-mail: ignacio.guajardo_m@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	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	4
2.4. indica el tiempo que demoró en obtener la password	5
2.5. descifra el contenido capturado	6
2.6. describe como obtiene la url de donde descargar el archivo	7
3. Desarrollo (PASO 2)	7
3.1. indica script para modificar diccionario original	7
3.2. cantidad de passwords finales que contiene rockyou_mod.dic	8
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	11
4.4. identifica nomenclatura del output	12
4.5. obtiene contraseña con aircrack-ng	13
4.6. identifica y modifica parámetros solicitados por pycrack	14
4.7. obtiene contraseña con pycrack	16

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 iniciar el proceso, se procede a configurar nuestro dispositivo de red en modo monitor empleando la herramienta Aircrack. Luego, ejecutamos el comando "sudo airodump-ng (con el identificador de nuestra tarjeta)" para dar inicio a la exploración de las redes detectadas. A partir de este análisis, se recopilan los datos representados en la figura 1.

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

Dentro de esta información, destaca la red con la dirección MAC B0:48:7A:XX:XX:XX, la cual es la única en la lista que utiliza encriptación WEP. Es importante señalar que el uso de WEP es obsoleto en las redes modernas, las cuales han migrado a estándares de seguridad como WPA o WPA2. Además, el profesor confirmó que esta red específica corresponde al laboratorio.

Nuestra búsqueda se centra en una red cuyo nombre es "WEP", y la característica distintiva que la diferencia de las demás redes en el listado es el campo ".ENC", que indica el tipo de encriptación utilizado. WEP es considerablemente menos seguro que WPA y WPA2, ya que

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

2 DESARROLLO (PASO 1)

```

Ignacio@Ignacio-Modern-14-B5M: ~
$ sudo airodump-ng -r wlan0mon -w cap
open failed: No such file or directory
Ignacio@Ignacio-Modern-14-B5M: ~
$ sudo airodump-ng wlan0mon -w cap
10:46:55 Created capture file 'cap-01.cap'.

CH 10 [ Elapsed: 17 mins ] [ 2023-10-20 11:04 ] [ WPA handshake: 98:1F:C1:11:86:B9 ]

BSSID PWR Beacons #Data, #/s CH MB ENC CIPHER AUTH ESSID
1E:82:7A:2C:F7:21 -63 178 36 0 11 130 WPA2 CCMP PSK POCO X3 Pro
80:1F:8C:E2:14:A3 -11 521 0 0 11 130 WPA3 CCMP OWE <length>: 0-
80:1F:8C:E2:14:A3 -10 595 270 0 11 130 OPM Alumnos-UDP
80:1F:8C:E2:14:A3 -11 615 4 0 11 130 WPA3 CCMP OWE <length>: 0-
80:1F:8C:E2:14:A3 -11 685 0 0 11 130 OPM VTP-UDP
80:1F:8C:E2:14:A3 -12 589 0 0 11 130 WPA3 CCMP SAE Sala Híbrida-UDP
80:1F:8C:E2:14:A3 -11 597 0 0 11 130 WPA2 CCMP CT Administrativos-UDP
80:1F:8C:E2:14:A3 -12 588 487 0 11 130 WPA3 CCMP OWE _owetr_Alumnos-UDP1993294148
80:1F:8C:E2:14:A3 -11 587 0 0 11 130 OPM Invitados-UDP
58:EF:68:47:59:C8 -9 400 39 0 6 130 WPA2 CCMP PSK cableadaTelenatica
58:EF:68:47:59:C8 -12 418 917 0 6 130 OPM cableadaTelenatica-Invitado
CC:ED:DC:1C:0E:78 -16 419 7 0 11 130 WPA2 CCMP PSK JPablo
CC:D4:A1:D7:81:D0 -16 510 11 0 11 130 WPA2 CCMP PSK HUAMEI_B2368-0781D0
84:08:1B:C6:83:E5 -18 266 0 0 2 195 WPA2 CCMP PSK FAMILIAGL_EXT
84:08:1B:C6:83:E5 -18 391 0 0 2 195 WPA2 CCMP PSK <length>: 0-
84:1C:30:85:EA:07 -20 356 0 0 10 130 WPA2 CCMP PSK ZTE_B5EA07
C3:85:E2:83:09:42 -21 136 0 0 1 130 WPA2 CCMP PSK CAPH
80:1F:8C:E8:E8:B4 -22 26 33 0 6 130 WPA3 CCMP OWE _owetr_Alumnos-UDP1993294148
1C:80:44:C4:40:F9 -24 18 18 0 11 130 WPA2 CCMP PSK MARTINO
48:13:43:23:80:D9 -25 66 1 0 11 130 WPA2 CCMP PSK VTR-2078881
CC:ED:DC:1C:0E:78 -25 58 0 0 6 130 WPA2 CCMP PSK movistar2_4GH2_98F122
CC:ED:DC:1C:0E:78 -25 9 0 0 11 130 WPA2 CCMP PSK Huelmo
AC:1F:8C:10:60:60 -25 86 3 0 11 130 WPA2 CCMP PSK VTR-8492879
E6:AB:89:1C:85:38 -26 33 5 0 11 130 WPA2 CCMP PSK ELL
AC:97:33:AA:2C:21 -28 65 0 0 1 130 WPA2 CCMP PSK Status deoa
AC:1F:8C:48:0A:D9 -27 61 0 0 6 130 WPA2 CCMP PSK VTR-9510941
AC:1F:8C:20:0C:08 -27 15 1 0 11 130 WPA2 CCMP PSK VTR-217650
7C:0B:96:43:F1:0F -27 19 0 0 11 130 WPA2 CCMP PSK Expedientes
80:1F:8C:E1:82:05 -27 160 0 0 11 130 OPM VIP-UDP
38:1B:00:1F:AC:71 -27 70 1 0 11 130 WPA2 CCMP PSK movistar_ACT1
18:35:D1:48:18:39 -27 7 0 0 1 130 WPA2 CCMP PSK VTR-5376275
80:1F:8C:E1:82:05 -28 142 3 0 11 130 WPA2 CCMP CT Administrativos-UDP
80:1F:8C:E1:82:04 -28 167 0 0 11 130 WPA3 CCMP OWE _owetr_Alumnos-UDP1993294148
EC:08:68:1C:D2:12 -28 184 0 0 2 270 WPA2 CCMP PSK CLINICA_VISITAS
14:CC:0B:18:38:35 -28 177 9 0 6 270 WPA2 CCMP PSK JPablo_EXT
46:4B:89:1F:66:AA -29 108 9 0 11 130 WPA2 CCMP PSK Depto 508
E4:4B:89:1F:66:AA -29 13 1 0 11 130 WPA2 CCMP PSK Rhyone
E4:4B:89:1F:66:AA -29 33 1 0 11 130 WPA2 CCMP PSK Juan Pablo
9C:90:7E:22:19:9D -29 142 0 0 8 130 WPA2 CCMP PSK Kata_rep
C3:86:27:4F:48:86 -30 0 0 11 130 WPA2 CCMP PSK HALCAVAGA
80:1F:8C:E1:82:05 -30 71 7 0 6 130 WPA3 CCMP PSK HUAMEI_B010_V044

```

Figura 1: Captura airodump

```

E6:AB:89:1C:85:38 -85 0 0 0 11 -1 <length>: 0
B0:48:7A:D2:DD:74 -44 71 553 4 3 54e WEP WEP WEP
B0:1F:8C:E0:F8:86 -24 2 0 0 6 130 WPA3 CCMP OWE <length>: 0

```

Figura 2:

su método de cifrado es más simple. En el caso de WEP, la única medida de seguridad es el cifrado del tráfico de datos. WEP utiliza un vector de inicialización para generar sus claves de cifrado, y este vector aumenta secuencialmente cada vez que se transmiten paquetes de datos. Por lo tanto, cuanto más paquetes se envíen, mayor es la probabilidad de detectar colisiones y, potencialmente, descifrar la clave de encriptación.

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

El método utilizado para obtener la clave de cifrado se conoce como el "Ataque de Cumpleaños", el cual busca encontrar la clave de cifrado mediante la detección de colisiones entre los paquetes de datos capturados. Este ataque se basa en un concepto matemático conocido como la "Paradoja de los Cumpleaños". Esta paradoja establece que si hay 23 personas en una habitación, existe una probabilidad del 50 % de que al menos dos de ellas compartan la misma fecha de cumpleaños. La probabilidad de colisión aumenta significativamente a más del 97 % cuando hay 50 personas presentes.

Para calcular la probabilidad de colisión en esta paradoja, se emplea la siguiente fórmula:

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

$$P(\text{colisión}) = 1 - \prod_{k=1}^{n-1} \left(1 - \frac{k}{n}\right)$$

Figura 3:

Donde n representa la cantidad de combinaciones posibles, en el caso de los cumpleaños nos $n = 365$, por lo tanto si se tienen 50 personas

$$P(\text{colisión}) = 1 - \prod_{k=1}^{50-1} \left(1 - \frac{k}{365}\right) = 97\%$$

Figura 4:

Ahora si cambiamos N a 2^{24} y probamos con 5000 elementos:

$$P(\text{colisión}) = 1 - \prod_{k=1}^{5000-1} \left(1 - \frac{k}{365}\right) = 99,7\%$$

Figura 5:

observamos que en este punto la probabilidad de encontrar una colision se acerca al 100 %

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

Para obtener la contraseña, el primer paso es capturar los paquetes de datos. En el paso anterior, hemos obtenido el BSSID de la señal, que en este caso es "B0:48:7A:D2:DD:74", y hemos identificado que la señal utiliza el canal 3. Ambos de estos datos son esenciales para interceptar el tráfico de la red. Para llevar a cabo la captura del tráfico, utilizamos el siguiente comando: `.airodump-ng -c 3 -bssid B0:48:7A:D2:DD:74 -w wep` (identificador de nuestra tarjeta)", donde los parametros son los siguientes:

- c = Canal de operacion
- bssid = MAC del dispositivo
- w = Identidficador del dispositivo de red

2.4 indica el tiempo que demoró en obtener la password 2 DESARROLLO (PASO 1)

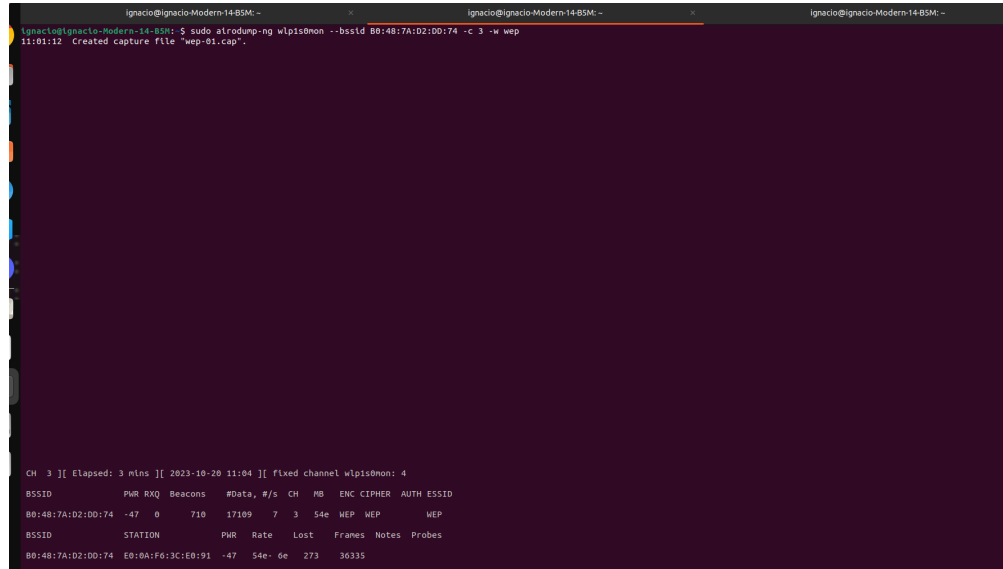


Figura 6: Captura de frames

Una vez que la captura ha sido iniciada, se espera a acumular una cantidad sustancial de "frames" de datos, en este caso, 17.109, y luego se detiene la captura. Estos paquetes capturados se utilizarán en el proceso de ataque para tratar de recuperar la clave de cifrado WEP.

2.4. indica el tiempo que demoró en obtener la password

Es importante notar que en las capturas los contenidos se encuentran cifrados.

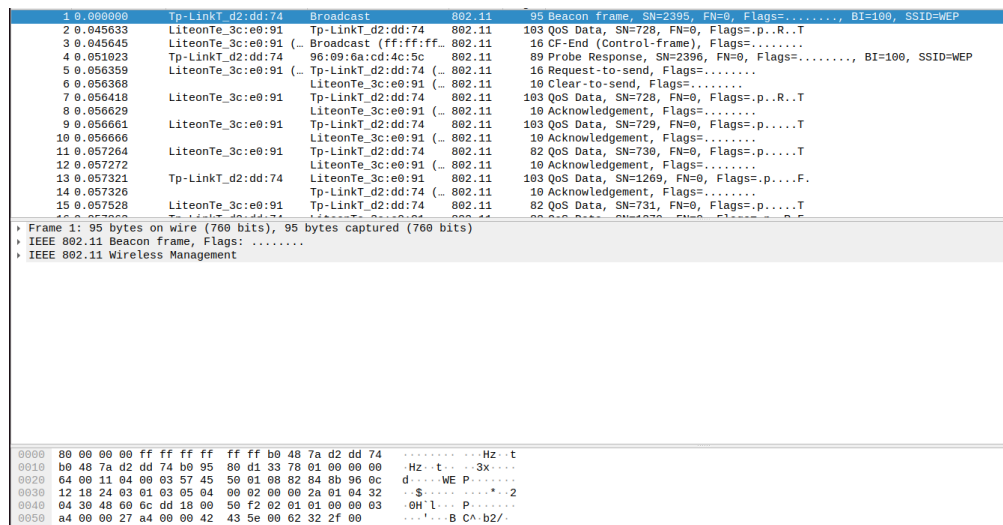


Figura 7: Captura cifrada

Para descifrar la clave de cifrado, se utilizará el comando aircrack-ng junto con la información de la red objetivo y el archivo de captura. En este caso, asumiendo que el archivo de captura se llama "wep.cap" que la dirección MAC de la red objetivo es "B0:48:7A:D2:DD:74", el comando se vería de la siguiente manera: "time aircrack-ng -b B0:48:7A:D2:DD:74 wep.cap"

```
ignacio@ignacio-Modern-14-B5M:~$ time aircrack-ng -b B0:48:7A:D2:DD:74 wep-01.cap
Reading packets, please wait...
Opening wep-01.cap
Read 66632 packets.

1 potential targets                                     Got 14505 out of 10000 IVsStarting PTW attack with 14505 ivs.

Attack will be restarted every 5000 captured ivs.

Aircrack-ng 1.6

[00:00:02] Tested 75924 keys (got 14505 IVs)

KB  depth  byte(vote)
0   1/ 14   12(19456) AC(19456) 45(19200) A4(18944) 80(18432) 86(18432) BC(18432) CA(18432) 07(18176) 3B(18176) 3E(18176) 2B(17920) 8B(17920) 39(17408) F4(17408) 3F(17152)
1   2/ 12   34(19712) BC(19712) 40(19456) 7B(18432) 80(18432) 6B(18432) DA(18176) F6(18176) 4B(18176) 66(17920) 77(17664) 9A(17664) 8B(17664) F4(17664) F0(17664) 05(17408)
2   0/ 1    56(24320) 80(20224) 86(19200) 00(18944) 93(18944) 15(18688) 73(18688) BF(18688) 80(18432) 30(18432) 89(18432) 2F(18176) 3F(18176) 60(17920) 05(17664) 2D(17664)
3   0/ 11   78(20992) 8B(19200) 35(18944) 79(18688) C0(18688) 00(18688) 24(18688) 55(18432) 7A(17920) 37(17920) 3A(17920) 89(17664) 8B(17664) AB(17408) AB(17408) C1(17408)
4   34/ 42  0E(16640) 3B(16384) 3F(16384) 60(16384) 76(16384) 77(16384) 9C(16384) A4(16384) AB(16384) AF(16384) C1(16384) CF(16384) D0(16384) EA(16384) EC(16384) F4(16384)

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

real    0m2.519s
user    0m2.501s
sys     0m0.020s
ignacio@ignacio-Modern-14-B5M:~$
```

Figura 8: Decifrado de clave

Es interesante notar que, en este caso, el comando 'aircrack-ng' logró descifrar correctamente la contraseña WEP, que resultó ser "12:34:56:78:90". Además, es interesante que el proceso de descifrado se completara en un tiempo tan breve, en menos de 5 segundos. Este resultado resalta la debilidad de la encriptación WEP y la facilidad con la que puede ser comprometida en comparación con estándares de seguridad más fuertes como WPA o WPA2.

2.5. descifra el contenido capturado

Para descifrar el contenido de la captura, utilizamos el comando airdecap-ng, suministrando la clave WEP "12:34:56:78:90" el archivo de captura "wep-01.cap".

```
ignacio@ignacio-Modern-14-B5M:~$ sudo airdecap-ng -w 12:34:56:78:90 wep-01.cap
Total number of stations seen      1
Total number of packets read      171783
Total number of WEP data packets  39551
Total number of WPA data packets   0
Number of plaintext data packets   0
Number of decrypted WEP packets    39551
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
ignacio@ignacio-Modern-14-B5M:~$
```

Figura 9:

La ejecución exitosa de este comando resulta en el descifrado del contenido de la captura, como se ilustra en la imagen mostrada a continuación.

2.6 describe como obtiene la url de donde descargar el archivo DESARROLLO (PASO 2)

8 0.014481	192.168.11.15	192.168.11.1	ICMP	54 Echo (ping) request	id=0x0005, seq=12327/10032, ttl=64 (reply in 11)
9 0.014826	192.168.11.1	192.168.11.15	ICMP	54 Echo (ping) reply	id=0x0005, seq=12326/9776, ttl=64 (request in 5)
10 0.017374	192.168.11.15	192.168.11.1	ICMP	54 Echo (ping) request	id=0x0005, seq=12328/10288, ttl=64 (reply in 13)
11 0.017427	192.168.11.1	192.168.11.15	ICMP	54 Echo (ping) reply	id=0x0005, seq=12327/10032, ttl=64 (request in 8)
12 0.017733	192.168.11.15	192.168.11.1	ICMP	54 Echo (ping) request	id=0x0005, seq=12329/10544, ttl=64 (reply in 14)
13 0.020050	192.168.11.1	192.168.11.15	ICMP	54 Echo (ping) reply	id=0x0005, seq=12328/10288, ttl=64 (request in 10)
14 0.022956	192.168.11.1	192.168.11.15	ICMP	54 Echo (ping) reply	id=0x0005, seq=12329/10544, ttl=64 (request in 12)
15 0.023217	192.168.11.15	192.168.11.1	ICMP	54 Echo (ping) request	id=0x0005, seq=12330/10800, ttl=64 (reply in 17)

Ethernet II, Src: Tp-LinkT_02:dd:74 (b0:48:7a:d2:dd:74), Dst: LiteonTe_3c:e0:91 (e0:0a:f6:3c:e0:91)

Internet Protocol Version 4, Src: 192.168.11.1, Dst: 192.168.11.15

Internet Control Message Protocol

Type: 0 (Echo (ping) reply)

Code: 0

Checksum: 0xba8b [correct]

[Checksum Status: Good]

Identifier (BE): 5 (0x0005)

Identifier (LE): 1280 (0x0500)

Sequence Number (BE): 12326 (0x3026)

Sequence Number (LE): 9776 (0x2630)

[Request frame: 5]

[Response time: 2,931 ms]

Data (12 bytes)

Data: 6269742e6c792f77061325f

[Length: 12]

0000 e0 0a f6 3c e0 91 b0 48 7a d2 dd 74 08 00 45 00 ...<...H z...t...E...

0010 00 28 b4 a8 00 00 40 01 2e cc c0 a8 0b 01 c0 a8 ...(.@...

0020 0b 0f 00 00 ba 8a 00 05 30 27 62 69 74 2e 6c 79 0'bit.ly

0030 2f 77 70 61 32 5f /wpa2

Figura 10:

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

Para obtener la URL, basta con observar cualquiera de los paquetes enviados por el dispositivo, y en la imagen siguiente se puede ver cómo se encuentra el enlace dentro de los contenidos del paquete. Ahora, una vez que hemos vulnerado los paquetes de captura, podemos examinar la información que está contenida en cada uno de ellos.

0000	e0 0a f6 3c e0 91 b0 48 7a d2 dd 74 08 00 45 00	...<...H z...t...E...
0010	00 28 b4 a8 00 00 40 01 2e cc c0 a8 0b 01 c0 a8	.(....@...
0020	0b 0f 00 00 ba 8a 00 05 30 27 62 69 74 2e 6c 79 0'bit.ly
0030	2f 77 70 61 32 5f	/wpa2

Figura 11:

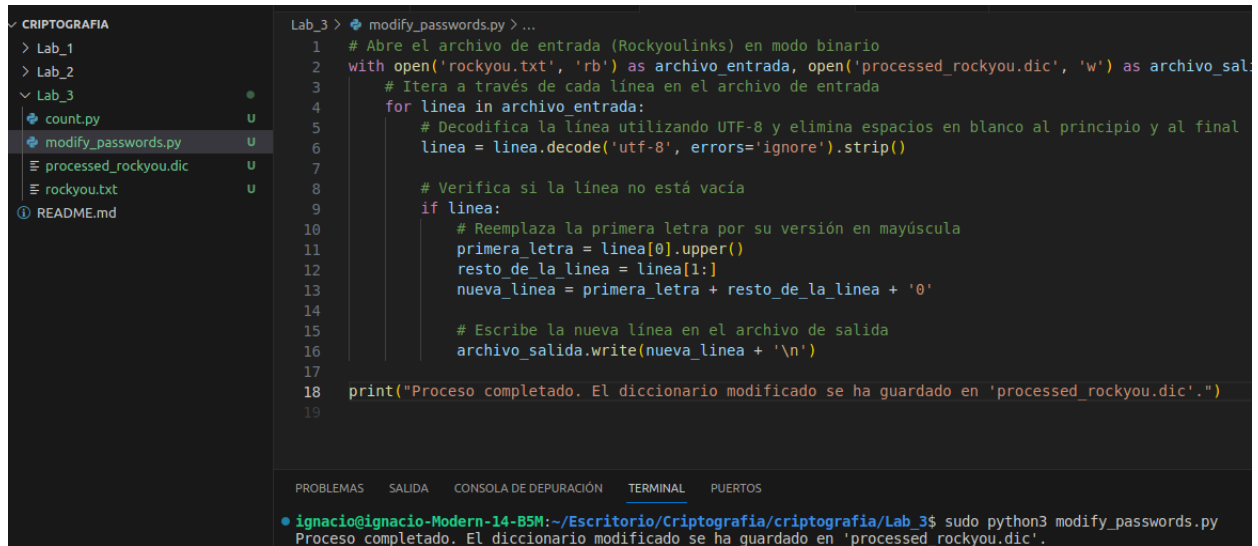
Si observamos detenidamente, dentro del contenido se encuentra la URL bit.ly/wpa2.. Si accedemos a este enlace, encontraremos el paquete compartido por el informante.

3. Desarrollo (PASO 2)

3.1. indica script para modificar diccionario original

En primer lugar, se procede a la descarga del archivo que contiene el diccionario rock-you.txt”, el cual contiene un listado de contraseñas. Para su posterior modificación, se creó el siguiente script. Sin embargo, antes de llevar a cabo cualquier modificación en el archivo rockyou.txt”, se requirió realizar dos pasos preliminares. En primer lugar, se identificaron errores de codificación en el archivo original, los cuales se corrigieron mediante la conversión a formato UTF-8. Posteriormente, se ejecutó el código en Python destinado a la modificación del archivo, y los resultados de esta operación se guardaron en un nuevo archivo denominado ”processed_rockyou.dic”.

3.2 cantidad de passwords finales que contiene rockyou_mod.dicDESARROLLO (PASO 2)



```
Lab_3 > modify_passwords.py > ...
1 # Abre el archivo de entrada (Rockyoulinks) en modo binario
2 with open('rockyou.txt', 'rb') as archivo_entrada, open('processed_rockyou.dic', 'w') as archivo_salida:
3     # Itera a través de cada línea en el archivo de entrada
4     for linea in archivo_entrada:
5         # Decodifica la línea utilizando UTF-8 y elimina espacios en blanco al principio y al final
6         linea = linea.decode('utf-8', errors='ignore').strip()
7
8         # Verifica si la línea no está vacía
9         if linea:
10            # Reemplaza la primera letra por su versión en mayúscula
11            primera_letra = linea[0].upper()
12            resto_de_la_linea = linea[1:]
13            nueva_linea = primera_letra + resto_de_la_linea + '\n'
14
15            # Escribe la nueva línea en el archivo de salida
16            archivo_salida.write(nueva_linea + '\n')
17
18 print("Proceso completado. El diccionario modificado se ha guardado en 'processed_rockyou.dic'.")
19
```

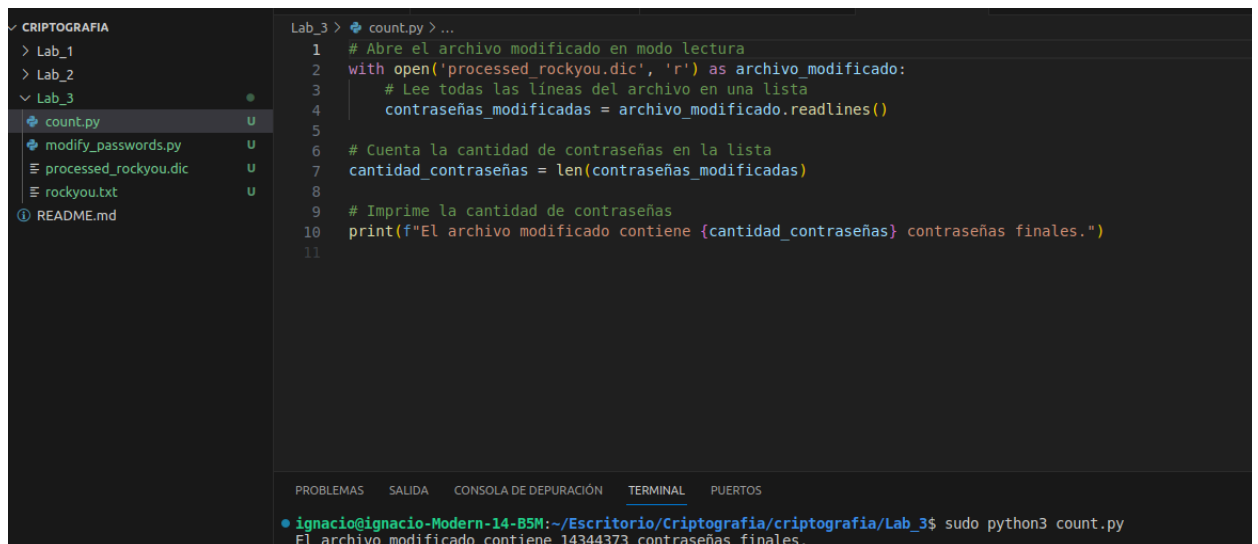
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

● ignacio@ignacio-Modern-14-B5M:~/Escritorio/Criptografia/criptografia/Lab_3\$ sudo python3 modify_passwords.py
Proceso completado. El diccionario modificado se ha guardado en 'processed_rockyou.dic'.

Figura 12: Script modificador de contraseñas

3.2. cantidad de passwords finales que contiene rockyou_mod.dic

El archivo resultante contiene un total de 14.344.373 passwords. Para esto se hizo un script sencillo el cual cuenta las líneas del archivo.

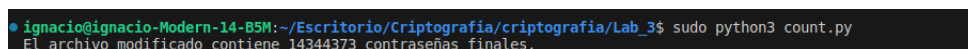


```
Lab_3 > count.py > ...
1 # Abre el archivo modificado en modo lectura
2 with open('processed_rockyou.dic', 'r') as archivo_modificado:
3     # Lee todas las líneas del archivo en una lista
4     contraseñas_modificadas = archivo_modificado.readlines()
5
6 # Cuenta la cantidad de contraseñas en la lista
7 cantidad_contraseñas = len(contraseñas_modificadas)
8
9 # Imprime la cantidad de contraseñas
10 print(f"El archivo modificado contiene {cantidad_contraseñas} contraseñas finales.")
11
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

● ignacio@ignacio-Modern-14-B5M:~/Escritorio/Criptografia/criptografia/Lab_3\$ sudo python3 count.py
El archivo modificado contiene 14344373 contraseñas finales.

Figura 13: Script que cuenta contraseñas



```
● ignacio@ignacio-Modern-14-B5M:~/Escritorio/Criptografia/criptografia/Lab_3$ sudo python3 count.py  
El archivo modificado contiene 14344373 contraseñas finales.
```

Figura 14: Cantidad de líneas

4. Desarrollo (Paso 3)

En el paso 1, accedemos al enlace proporcionado por el informante, que redirige a la siguiente URL: <https://www.cloudshark.org/captures/b5b39e1c51eb>. Esta URL corresponde a una captura de Wireshark que ha sido descargada. Sin embargo, se advierte que también esta captura se encuentra cifrada, por lo que será necesario descifrarla.

Para descifrar la contraseña, haremos uso de tres herramientas: aircrack-ng, pycrack y hashcat. Aircrack-ng ya está instalado en el sistema. Para instalar Hashcat, se empleará el siguiente comando: "sudo apt-get install hashcat"

4.1. obtiene contraseña con hashcat con potfile

Para poder utilizar Hashcat con archivos capturados en formato .cap, es necesario convertirlos a un formato aceptable. Afortunadamente, Hashcat ofrece un sitio web que permite realizar esta conversión.

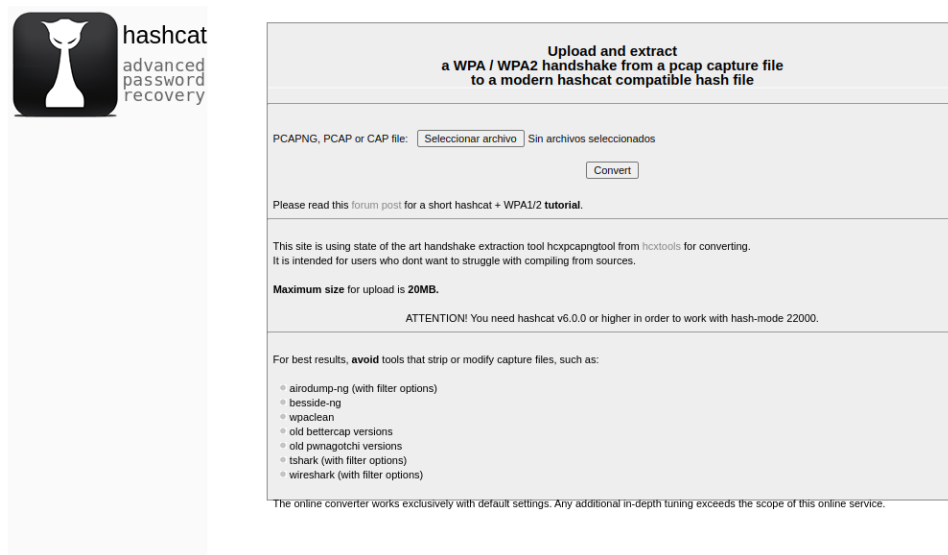


Figura 15:

Una vez que se realizó la conversión del archivo capturado, se ejecutó el siguiente comando utilizando Hashcat: "sudo hashcat -m 22000 -a 0 handshake.hc22000 rockyou.dic"

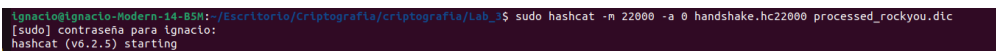


Figura 16:

En este contexto, los parámetros utilizados fueron los siguientes:

- sudo hashcat: Hashcat fue iniciado con privilegios de superusuario para llevar a cabo la operación.

- -m 22000: Se seleccionó el modo 22000 de Hashcat, diseñado específicamente para hashes de contraseñas de Wi-Fi WPA/WPA2.
- -a 0: Se optó por un ataque de fuerza bruta (valor 0 en el parámetro -a). Hashcat procedió a probar todas las combinaciones posibles de la lista de contraseñas.
- handshake.hc22000: Este archivo contenía el hash de la clave a descifrar, capturado durante el proceso anterior.
- rockyou.dic: El archivo utilizado como lista de contraseñas comunes, que Hashcat empleó para intentar descifrar el hash.

El potfile es el archivo donde se almacenan las contraseñas previamente descifradas y este se encuentra activado por defecto en Hashcat. En otras palabras, Hashcat guarda automáticamente las contraseñas que ha descifrado con éxito en el potfile. Esta característica permite que las contraseñas descriptadas estén disponibles para su uso posterior o referencia, lo que puede ser útil para diversos fines, como el análisis de seguridad o la auditoría de contraseñas.

4.2. identifica nomenclatura del output

Después de ejecutar el comando anterior, se obtiene la salida que se muestra en la imagen adjunta. En esta salida, se pueden observar múltiples campos que detallan el procedimiento realizado y su estado final.

```
Dictionary cache built:
* Filename.: processed_rockyou.dic
* Passwords.: 14344373
* Bytes.....: 154263471
* Keyspace.: 14344366
* Runtime...: 1 sec

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

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 22000 (WPA-PBKDF2-PMKID+EAPOL)
Hash.Target.....: handshake.hc22000
Time.Started....: Sat Oct 21 13:04:11 2023 (1 sec)
Time.Estimated...: Sat Oct 21 13:04:12 2023 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (processed_rockyou.dic)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 6431 H/s (11.04ms) @ Accel:256 Loops:256 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests
Progress.....: 5646/14344366 (0.04%)
Rejected.....: 2574/5646 (45.59%)
Restore.Point....: 0/14344366 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: 1234567890 -> Pistons0
Hardware.Mon.#1...: Temp: 51c Util: 35%

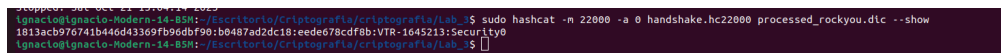
Started: Sat Oct 21 13:03:17 2023
Stopped: Sat Oct 21 13:04:14 2023
```

Figura 17: Candidato

Algunos de los campos destacables son:

- Hash Descifrado: Se encuentra arriba del listado, en orden se nos entrega el Hash, la MAC de destino, la MAC de origen, el SSID y la contraseña en texto plano, en este caso Security0, estos valores se pueden verificar al revisar la captura
- Status: indica si se pudo descifrar la contraseña, en este caso el valor es Cracked, indicando que si se logro
- Hash.Mode: Indica el paquete utilizado para el proceso en este caso 22000 (WPA)
- Time: .started indica el tiempo de inicio y .finished el tiempo de finalizacion
- Speed: Indica la velocidad a la que se realizo el proceso en hashes por segundo
- Progress: Indica cuantas contraseñas se probaron del total

Podemos verificar el potfile, utilizando el comando `sudo hashcat -m 22000 -a 0 handshake.hc22000 processed rockyou.dic --show`



```

ignacio@ignacio-Rodern-14-B5M: /usr/local/bin $ sudo hashcat -m 22000 -a 0 handshake.hc22000 processed_rockyou.dic --show
1813ac9976741b46d43369f996dbf90-b0487ad3dc181eede678cdf8b:VTR-1645213:Security0
ignacio@ignacio-Rodern-14-B5M: /usr/local/bin $

```

Figura 18:

Es importante destacar que si ejecutas nuevamente la operación para descifrar contraseñas utilizando Hashcat y la contraseña ya se encuentra en el potfile (archivo donde se almacenan las contraseñas previamente descifradas), el proceso finalizará inmediatamente. Hashcat verificará si la contraseña en cuestión ya ha sido descifrada y almacenada en el potfile para evitar operaciones innecesarias. Esta optimización ahorra tiempo y recursos al evitar repetir el proceso de descifrado para contraseñas que ya se conocen.

4.3. obtiene contraseña con hashcat sin potfile

Si deseas evitar el uso de un potfile, simplemente debes añadir la opción `--potfile-disable` al comando previamente ejecutado. El comando resultante sería:

”`sudo hashcat -m 22000 -a 0 handshake.hc22000 rockyou.dic --potfile-disable`”.

Al agregar la opción `--potfile-disable`, Hashcat no utilizará un potfile para verificar contraseñas previamente descifradas y permitirá que el proceso continúe sin tener en cuenta si las contraseñas ya se encuentran en el potfile o no. Ten en cuenta que esta opción puede resultar en un mayor uso de recursos y tiempo, ya que no se realizarán las optimizaciones de descifrado basadas en el potfile.

```
Dictionary cache hit:  
* Filename.: processed_rokyou.dic  
* Passwords.: 14344366  
* Bytes.....: 154263471  
* Keyspace...: 14344366  
  
1813acb976741b446d43369fb96dbf90:b0487ad2dc18:eede678cdf8b:VTR-1645213:Security0  
  
Session.....: hashcat  
Status.....: Cracked  
Hash.Mode.....: 22000 (WPA-PBKDF2-PMKID+EAPO)  
Hash.Target.....: handshake.hc22000  
Time.Started....: Sat Oct 21 13:21:10 2023 (0 secs)  
Time.Estimated...: Sat Oct 21 13:21:10 2023 (0 secs)  
Kernel.Feature...: Pure Kernel  
Guess.Base.....: File (processed_rokyou.dic)  
Guess.Queue.....: 1/1 (100.00%)  
Speed.#1.....: 15483 H/s (12.03ms) @ Accel:128 Loops:512 Thr:1 Vec:8  
Recovered.....: 1/1 (100.00%) Digests  
Progress.....: 5646/14344366 (0.04%)  
Rejected.....: 2574/5646 (45.59%)  
Restore.Point...: 2986/14344366 (0.02%)  
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1  
Candidate.Engine.: Device Generator  
Candidates.#1...: Crazygirl0 -> Plstons0  
Hardware.Mon.#1..: Temp: 40c Util: 22%  
  
Started: Sat Oct 21 13:21:08 2023  
Stopped: Sat Oct 21 13:21:12 2023  
lgnacio@lgnacio-Modern-14-R5M:~/Escritorio/Criptografia/cryptografia/Lab_3$
```

Figura 19:

Es importante destacar que si ejecutas la operación nuevamente con la opción `-potfile-disable`, los resultados pueden ser idénticos a los obtenidos previamente, ya que la operación se ejecuta de nuevo sin tener en cuenta el potfile. Esto significa que Hashcat volverá a intentar descifrar las contraseñas incluso si ya se encuentran en el potfile.

La opción `-potfile-disable` elimina la verificación del potfile y permite que Hashcat intente descifrar todas las contraseñas nuevamente, sin importar si ya han sido descifradas en el pasado. Esta opción puede ser útil en casos en los que se quiera ejecutar el proceso de descifrado desde cero o para realizar pruebas adicionales.

Es importante tener en cuenta que, al eliminar el uso del potfile, el proceso podría llevar más tiempo y consumir más recursos en comparación con una ejecución que lo tenga en cuenta para evitar duplicados.

4.4. identifica nomenclatura del output

La nomenclatura utilizada en Hashcat es la misma que se muestra y se explica en relación al potfile. Esta nomenclatura estandarizada asegura que los resultados y las contraseñas descifradas se almacenen y gestionen de manera coherente, lo que facilita la identificación y el seguimiento de las contraseñas que ya han sido descifradas y se encuentran en el potfile. Esta consistencia es importante para mantener la integridad de los resultados y optimizar el proceso de descifrado.

4.5. obtiene contraseña con aircrack-ng

```

Type: IEEE 802.11 Management Response (0x00000000)
  Frame Control Field: 0x1000
  .0000 0001 0100 0000 = Duration: 320 microseconds
  Receiver address: ee:de:67:8c:df:8b (ee:de:67:8c:df:8b)
  Destination address: ee:de:67:8c:df:8b (ee:de:67:8c:df:8b)
  Transmitter address: Tp-LinkT_d2:dc:18 (b0:48:7a:d2:dc:18)
  Source address: Tp-LinkT_d2:dc:18 (b0:48:7a:d2:dc:18)
  BSS Id: Tp-LinkT_d2:dc:18 (b0:48:7a:d2:dc:18)
  .... 0000 = Fragment number: 0
  0100 1010 0000 .... = Sequence number: 1184
  IEEE 802.11 Wireless Management

0000 10 00 40 01 ee de 67 8c df 8b b0 48 7a d2 dc 18 ..@...g...Hz...
0010 b0 48 7a d2 dc 18 00 4a 31 04 00 00 01 c0 01 08 .Hz...J 1.....
0020 82 84 8b 96 0c 12 18 24 32 04 30 48 60 6c 7f 08 .....$ 2.0H`l..
0030 04 00 00 00 01 00 01 40 6e 12 00 00 02 10 01 01 .....@ n.....
0040 ff ff 12 16 18 26 28 28 2c 2e 30 38 dd 18 00 50 .....&(( ,.08...P
0050 f2 02 01 01 81 00 03 a4 00 00 27 a4 00 00 42 43 .....BC
0060 5e 00 62 32 2f 00 ^..b2/..

```

Figura 20: bssid

Para obtener la contraseña, se utilizará el siguiente comando con aircrack-ng: aircrack-ng -a2 -b b0:48:7a:d2:dc:18 -w processed_rockyou.dic handshake.pcap A continuación, se describen los parámetros utilizados en este comando:

- -a2: Este parámetro indica que se utilizará un ataque de diccionario, lo que significa que se probarán todas las contraseñas contenidas en el archivo de diccionario especificado
- -b b0:48:7a:d2:dc:18: Aquí se proporciona la dirección MAC de la red Wi-Fi objetivo, identificada como "b0:48:7a:d2:dc:18".
- -w processed: Esta opción especifica el nombre del archivo de salida donde se guardarán los resultados del proceso de descifrado.
- rockyou.dic: Es el archivo de diccionario que se utilizará para probar las contraseñas. Hashcat intentará descifrar la clave utilizando las contraseñas contenidas en este archivo.
- handshake.pcap: Es el archivo de handshake descargado previamente y que contiene la información necesaria para intentar descifrar la contraseña de la red Wi-Fi.

Una vez ejecutado el comando obtenemos los siguientes resultados:

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

```

    AirCrack-ng 1.6

    [00:00:00] 3002/9373846 keys tested (12707.75 k/s)

    Time left: 12 minutes, 17 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
  
```

Figura 21: Resultados Aircrack

4.6. identifica y modifica parámetros solicitados por pycrack

<https://github.com/nogilnick/PyCrack/blob/master/pywd.py> Para utilizar PyCrack, debes comenzar descargando o clonando el repositorio desde GitHub (<https://github.com/nogilnick/PyCrack>). En este caso descargaremos y modificaremos el archivo `pywd.py` con los parametros respectivos donde:

[illegible]

Figura 22: Parametros Pycrack

Estos seran reemplazados por los datos que se encuentran en el Handshake quedando:

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

[illegible]

Figura 23: Parámetros Pycrack modificados

- [illegible]

Los datos se obtuvieron copiando los frames completos de las capturas del HandShake, específicamente los frames 2, 3 y 4. Un detalle importante es que se reemplaza el valor del MIC por 0 en estos frames.

Además, se lleva a cabo la desactivación de la función `RunTest()` se inserta el diccionario previamente creado `"processed_rockyou.dic"`. Esta modificación permite que el programa utilice el diccionario personalizado para intentar descifrar las contraseñas, en lugar de realizar pruebas de fuerza bruta con un conjunto de contraseñas predeterminadas.

4.7. obtiene contraseña con pycrack



```
ignacio@ignacio-Modern-14-B5M:~/Escritorio/Criptografia/cryptografia/Lab_3$ sudo python3 pywd.py
[sudo] contraseña para ignacio:
!!!Password Found!!!
Desired MIC1:      1813acb976741b446d43369fb96dbf90
Computed MIC1:     1813acb976741b446d43369fb96dbf90

Desired MIC2:      a349d01089960aa9f94b5857b0ea10c6
Computed MIC2:     a349d01089960aa9f94b5857b0ea10c6

Desired MIC2:      5cf0d63af458f13a83daa686df1f4067
Computed MIC2:     5cf0d63af458f13a83daa686df1f4067
Password:          Security0
ignacio@ignacio-Modern-14-B5M:~/Escritorio/Criptografia/cryptografia/Lab_3$
```

Figura 24: Resultados Pycrack

Obteniendo la password "Security0".

Conclusiones y comentarios

En conclusión, en esta experiencia de laboratorio se puso en práctica los conceptos aprendidos en experiencias anteriores, tales como la encriptación en paquetes ICMP y la decodificación en base64, así también como conceptos nuevos que se aprendieron en esta experiencia que fue interceptar paquetes de una red inalámbrica WEP, capturarlos y utilizarlos en conjunto a un diccionario de contraseñas para poder obtener la contraseña de la red. Se profundizó en las posibilidades para descifrar un hash, así como en los procesos que son necesarios para esto, nuevamente se mejoró el entendimiento de los ataques de fuerza bruta, utilizando diccionarios para encontrar contraseñas. Luego de explorar herramientas como Aircrack-ng, Hashcat y PyCrack, hemos adquirido una percepción más profunda sobre la relativa facilidad de llevar a cabo ataques en una red, no solo con el propósito de descifrar contraseñas, sino también para acceder al texto claro de la comunicación. Sería de gran valor realizar un análisis más exhaustivo de la lógica subyacente en cada uno de estos métodos para descifrar contraseñas, y comprender cuándo y cómo aplicar cada uno de ellos en diversas situaciones. Esto permitirá una toma de decisiones más informada y efectiva en la protección de redes y sistemas, y en la evaluación de su seguridad. Con un conocimiento más profundo de estas técnicas, se podrá diseñar e implementar estrategias de seguridad más robustas y mitigar los riesgos asociados a posibles ataques. Se cumplieron los objetivos de la experiencia en su totalidad, obteniendo resultados consistentes entre cada una de las actividades.