

Informe Laboratorio 3

Sección 2

Jonathan A. Cuitiño Mendoza
e-mail: jonathan.cuitino@mail.udp.cl

Mayo 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	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	11
4.3. obtiene contraseña con hashcat sin potfile	13
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	19
5. Conclusiones y comentarios	20

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)

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

Hay distintos puntos en base a los cuales se puede identificar la red buscada. Para esto, se debe analizar en las redes presentes en el ambiente en el que estamos.

2.1 identificar en qué se destaca la red del informante del res2o DESARROLLO (PASO 1)

```
(john@kali)-[~]
└─$ sudo airodump-ng wlan0mon
```

CH 7][Elapsed: 6 s][2023-10-17 09:50]

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC CIPHER	AUTH	ESSID
20:AA:4B:31:A2:D4	-74	0	0 0	1	130	WPA2 CCMP	PSK	OF-CCFI
CC:ED:DC:8A:F7:FF	-76	2	0 0	1	130	WPA2 CCMP	PSK	movistar2,4GHZ_8AF7FF
A4:97:33:A6:2C:21	-79	2	0 0	1	130	WPA2 CCMP	PSK	MOVISTAR_2C1F
B0:1F:8C:E2:14:A4	-61	4	0 0	1	130	WPA3 CCMP	OWE	<length: 0>
98:FC:11:86:B6:B9	-52	7	3 1	6	130	WPA2 CCMP	PSK	Telematica
F0:61:C0:E8:2F:81	-76	2	0 0	6	130	OPN		Invitados-UDP
B0:1F:8C:E1:B2:01	-67	2	0 0	11	130	OPN		Invitados-UDP
B0:1F:8C:E1:B2:04	-67	4	0 0	11	130	WPA3 CCMP	OWE	<length: 0>
B0:1F:8C:E1:B2:03	-67	3	0 0	11	130	OPN		Alumnos-UDP
78:29:ED:AF:C6:D7	-77	1	6 0	11	130	WPA2 CCMP	PSK	MOVISTARF
48:D3:43:33:B9:D9	-74	2	0 0	11	130	WPA2 CCMP	PSK	VTR-2078881
B0:1F:8C:E1:B2:07	-67	2	0 0	11	130	WPA2 CCMP	MGT	Administrativos-UDP
B0:1F:8C:E1:B2:06	-66	3	0 0	11	130	WPA3 CCMP	OWE	<length: 0>
B0:1F:8C:E1:B2:02	-67	4	0 0	11	130	WPA3 CCMP	OWE	<length: 0>
B0:1F:8C:E1:B2:00	-66	5	0 0	11	130	WPA3 CCMP	SAE	Sala Hibrida-UDP
B0:48:7A:D2:DD:74	-44	8	259 0	6	54e	WEP	WEP	WEP
84:D8:1B:C6:83:E9	-72	4	0 0	2	195	WPA2 CCMP	PSK	FAMILIAGL_EXT
8A:D8:1B:C6:83:E9	-66	5	0 0	2	195	WPA2 CCMP	PSK	<length: 0>
CC:D4:A1:D7:81:DD	-69	0	0 13	130	WPA2 CCMP	PSK	HUAWEI-B2368-D781DD	
CC:ED:DC:1C:0E:71	-65	4	0 0	13	130	WPA2 CCMP	PSK	JPablo
14:CC:20:E8:EB:35	-67	3	0 13	270	WPA2 CCMP	PSK	JPablo_EXT	
58:EF:68:47:59:C8	-69	5	0 0	6	130	OPN		cableadaTelematica-invitado
58:EF:68:47:59:C6	-67	6	1 0	6	130	WPA2 CCMP	PSK	cableadaTelematica
44:48:B9:41:A2:D8	-76	4	0 0	1	130	WPA2 CCMP	PSK	CECI
48:D3:43:BE:A7:19	-69	1	5 0	1	130	WPA2 CCMP	PSK	VTR-7335606
B0:1F:8C:E2:14:A6	-62	6	0 0	1	130	WPA3 CCMP	OWE	<length: 0>
B0:1F:8C:E2:14:A2	-63	3	0 0	1	130	WPA3 CCMP	OWE	<length: 0>
B0:1F:8C:E2:14:A1	-62	3	0 0	1	130	OPN		Invitados-UDP
B0:1F:8C:E2:14:A0	-63	6	0 0	1	130	WPA3 CCMP	SAE	Sala Hibrida-UDP
10:6F:3F:0E:74:A0	-70	4	0 0	1	270	WPA2 CCMP	PSK	LSA
40:0D:10:30:2A:59	-74	4	0 0	1	130	WPA2 CCMP	PSK	VTR-0744278
E4:AB:89:60:59:60	-77	4	0 0	1	130	WPA2 CCMP	PSK	DPTO 305
E4:AB:89:67:33:90	-72	5	0 0	1	130	WPA2 CCMP	PSK	Otakus depa
44:48:B9:4A:1C:F8	-74	2	0 0	1	130	WPA2 CCMP	PSK	Javiera
B0:1F:8C:E2:14:A7	-61	4	0 0	1	130	WPA2 CCMP	MGT	Administrativos-UDP
B0:1F:8C:E2:14:A5	-62	7	0 0	1	130	OPN		VIP-UDP
52:FB:78:35:DF:AF	-37	11	2 0	1	360	WPA2 CCMP	PSK	AndroidNato
B0:1F:8C:E2:14:A3	-63	4	0 0	1	130	OPN		Alumnos-UDP
AC:F8:CC:1D:60:60	-65	4	0 0	1	130	WPA2 CCMP	PSK	VTR-8492879
B4:1C:30:B5:EA:07	-58	9	3 1	10	130	WPA2 CCMP	PSK	ZTE_B5EA07

BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes
98:FC:11:86:B6:B9	96:DB:7D:AB:36:60	-52	0 - 1	3	18		
78:29:ED:AF:C6:D7	80:C5:F2:1F:50:53	-71	1e- 0	0	6		
B0:48:7A:D2:DD:74	B8:27:EB:35:AB:17	-47	54e-54e	40	260		
(not associated)	92:53:51:BB:8E:CE	-73	0 - 1	1	2		Alumnos-UDP
(not associated)	26:27:34:1E:D5:93	-76	0 - 1	0	1		
(not associated)	B2:06:0B:7A:B9:61	-71	0 - 1	0	1		
(not associated)	9A:D7:86:E4:EB:D3	-53	0 - 1	1	2		
(not associated)	DA:A1:19:CA:7B:40	-81	0 - 1	0	1		
(not associated)	54:88:0E:48:36:E8	-77	0 - 1	0	2		
(not associated)	DE:2C:B0:91:96:FC	-58	0 - 1	0	1		

Figura 1: Redes presentes al momento de la medición

Al analizar las redes presentes, se aprecia que hay una red que destaca sobre las demás por los siguientes factores:

- **Cantidad de paquetes enviados:** El campo *#Data* de la imagen precedente indica la cantidad de paquetes que se capturaron de esa red. Claramente tiene un trafico evidentemente mayor a las demás redes presentes.
- **Cifrado de la red:** Actualmente las mayorías de las redes inalámbricas estan cifradas con WPA2 O WPA3. Sin embargo, hay una red que esta cifrada con WEP.

- **Potencia de la señal:** Hay una señal que se esta recibiendo con mayor potencia que las demás (-44dB).
- **Velocidad de la red:** Todas las redes estan dentro de un rango de velocidades normal (130-360 MBps), esto se puede ver en el campo MB. Se aprecia de esta manera que hay una red con velocidad distinta a todas las redes presentes (54e).

Al analizar todos estos campos y factores anteriormente detallados, es evidente cual es la red del informante: **WEP**.

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

El birthday attack es un ataque de colisión por fuerza bruta que explota las matemáticas detrás del problema del cumpleaños en la teoría de la probabilidad. Este ataque puede utilizarse para abusar de la comunicación entre dos o más partes.

Matemáticas detrás del proceso: Dada una función f el objetivo del ataque es encontrar dos entradas diferentes x_1 , x_2 tal que $f(x_1) = f(x_2)$. Tal par x_1 , x_2 es llamado colisión. El método utilizado para encontrar una colisión es simplemente evaluar la función f para diferentes valores de entrada que pueden ser elegidos al azar o pseudo aleatoriamente hasta que el mismo resultado se encuentra más de una vez. Debido al problema del cumpleaños, este método puede ser bastante eficiente. En concreto, si una función $f(x)$ produce cualquiera de H salidas diferentes con igual probabilidad y H es suficientemente grande, entonces esperamos obtener un par de argumentos diferentes x_1 y x_2 con $f(x_1) = f(x_2)$ después de evaluar la función para aproximadamente $1,25 \cdot \sqrt{H}$ diferentes argumentos en promedio.

En nuestro caso, para determinar la cantidad de paquetes requeridos para obtener el passcode, se debe usar la siguiente formula.

$$n(p; H) \approx \sqrt{2H \ln \frac{1}{1-p}}$$

Figura 2: Calculo de paquetes requeridos

Donde:

- $n(p; H)$ Es el numero de paquetes requeridos para obtener la contraseña.
- p es la probabilidad de encontrar una colisión.
- H Conjunto de valores.

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

Para nuestro escenario, $p = 0.5$, dado que necesitamos el 50 % de probabilidad de éxito. La cantidad de valores posibles es 2 elevado a 24. Finalmente, al aplicar la formula recientemente expuesta se obtiene:

$$n(p;H) = 4822,67 \text{ paquetes.}$$

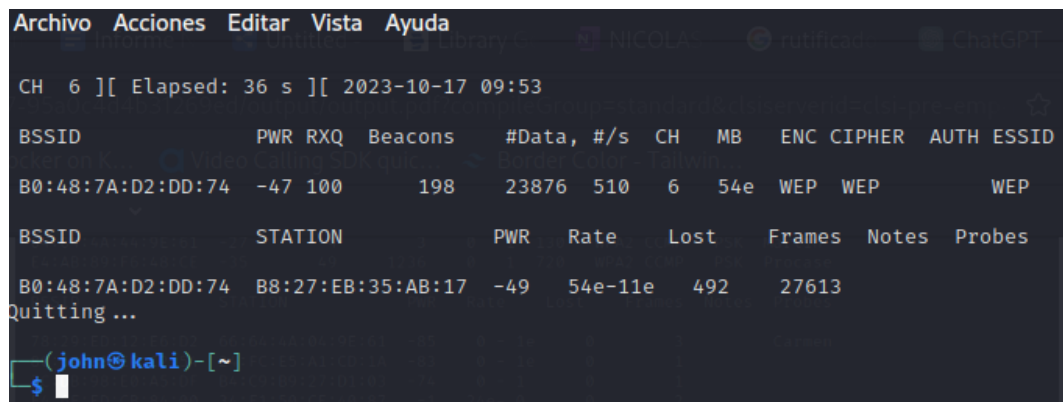
Finalmente, este valor es aproximado a 5000 paquetes capturados.

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

Para obtener la password se hace uso de la suite de **aircrack-ng**. El primer paso para obtener la contraseña fue identificar la red del informante, tal como se explicita en el punto 2.1. Una vez identificado el AP en cuestión, se utiliza su BSSID para capturar los paquetes relacionados con el AP, a través del siguiente comando:

```
sudo airodump-ng -c 6 --bssid B0:48:7A:D2:DD:74 -w archivo_cripto wlan0mon
```

Al ingresar esta instruccion por consola, comienza la captura de paquetes que sera guardado en el archivo con nombre especificado en el comando, posterior a la bandera -w, como sigue:



```
Archivo Acciones Editar Vista Ayuda
CH 6 ][ Elapsed: 36 s ][ 2023-10-17 09:53

BSSID          PWR RXQ Beacons  #Data, #/s  CH  MB  ENC CIPHER AUTH ESSID
B0:48:7A:D2:DD:74 -47 100   198    23876  510   6   54e  WEP  WEP   WEP   WEP

BSSID          STATION          PWR   Rate    Lost    Frames  Notes  Probes
B0:48:7A:D2:DD:74 B8:27:EB:35:AB:17 -49    54e-11e  492     27613

Quitting ...

(john@kali)-[~]
$
```

Figura 3: Captura de paquetes asociada al BSSID del AP del informante

Una vez capturados los paquetes, estos son analizados por aircrack en busca de la contraseña, mediante el comando:

```
sudo aircrack-ng -b B0:48:7A:D2:DD:74 archivo_cripto-01.cap
```

En este caso, como la red usa cifrado WEB, la contraseña se mostrará en forma de una cadena hexadecimal:

```

Aircrack-ng 1.7

[00:00:01] Tested 6340 keys (got 23882 IVs)

KB    depth  byte(vote)
0     0/ 1    12(35072) 54(30208) 59(29952) EB(29952) 49(29184) 4A(29184) 96(29184) C4(28928)
1     0/ 24    34(30976) 8B(30720) DB(29696) B8(29184) F5(28672) A3(28672) FE(28416) 46(28160)
2     2/ 12    56(29440) AD(29184) 3D(28416) 04(28416) 1E(28160) 8D(27904) C9(27904) 74(27648)
3     0/ 1     78(36608) 05(32000) E6(31744) FD(31488) 49(30976) BB(29696) 3B(29440) BC(29440)
4    20/ 23    D7(27392) 74(27136) A4(27136) BA(26880) D8(26880) 0D(26624) 25(26624) 33(26624)

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

```

Figura 4: Obtención de la contraseña en formato HEX

De estas forma, la clave utilizada para cifrar el trafico de la red, en formato HEX es: **12:34:56:78:90**.

2.4. indica el tiempo que demoró en obtener la password

El tiempo requerido para obtener la contraseña fue una vez conocidos los comandos a utilizar y como emplearlos es de al rededor de 5 o 10 minutos, como máximo. Sin embargo, se debe investigar como funciona cada comando, bandera y parametro, lo que implica un tiempo extra considerablemente mayor, debiendo emplear casi todo el bloque del laboratorio (1.5 hrs) aprendiendo como usar la suite de aircrack.

En cuanto al tiempo de ejecución del comando, este se puede obtener antecediendo el parametro time al comando anteriormente descrito. De esta forma se obtiene el tiempo real que se tarda el comando en descifrar la clave (2.11s):

```

Aircrack-ng 1.7

[00:00:02] Tested 6340 keys (got 23882 IVs)

KB    depth  byte(vote)
0     0/ 1    12(35072) 54(30208) 59(29952) EB(29952) 49(29184) 4A(29184) 96(29184) C4(28928)
1     0/ 24    34(30976) 8B(30720) DB(29696) B8(29184) F5(28672) A3(28672) FE(28416) 46(28160)
2     2/ 12    56(29440) AD(29184) 3D(28416) 04(28416) 1E(28160) 8D(27904) C9(27904) 74(27648)
3     0/ 1     78(36608) 05(32000) E6(31744) FD(31488) 49(30976) BB(29696) 3B(29440) BC(29440)
4    20/ 23    D7(27392) 74(27136) A4(27136) BA(26880) D8(26880) 0D(26624) 25(26624) 33(26624)

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

real    2,11s
user    0,00s
sys     0,00s
cpu     0%

```

Figura 5: Tiempo que tarda aircrack en descifrar la contraseña.

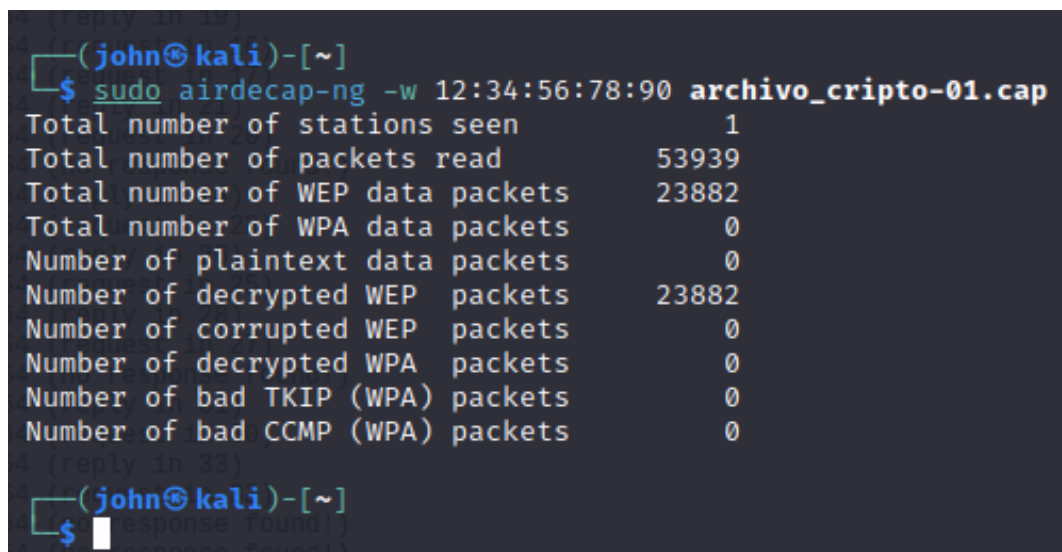
2.5. descifra el contenido capturado

Una vez realizado el paso anterior, se procede a descifrar el contenido capturado, esto haciendo uso del siguiente comando:

```
sudo airdecap-ng -w 12:34:56:78:90 archivo_cripto-01.cap
```

El uso de la herramienta **airdecap-ng**, de la suite de aircrack-ng se utiliza para intentar descifrar un archivo de captura que contiene tráfico de una red Wi-Fi protegida mediante cifrado WEP.

El resultado del comando dependerá de si la clave WEP proporcionada es correcta y puede descifrar con éxito el archivo de captura. Si la clave es correcta y coincide con la clave utilizada en la red Wi-Fi, el comando descifrará el archivo y generará un nuevo archivo con la extensión *.dec* que contendrá el tráfico de red en texto claro. Por ejemplo, si la clave es correcta, se creará un archivo llamado *archivo_cripto-01-dec.cap* que contendrá el tráfico descifrado en texto claro:



```
(john@kali)-[~]  
$ sudo airdecap-ng -w 12:34:56:78:90 archivo_cripto-01.cap  
Total number of stations seen          1  
Total number of packets read          53939  
Total number of WEP data packets      23882  
Total number of WPA data packets      0  
Number of plaintext data packets      0  
Number of decrypted WEP packets       23882  
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  
(john@kali)-[~]  
$
```

Figura 6: Generación del archivo -dec.cap para ser analizado en wireshark

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

Una vez generado el archivo *archivo_cripto-01-dec.cap*, este puede ser analizado vía wireshark, de la siguiente manera:

11	0.000070	192.168.11.3	192.168.11.1	ICMP	54 Echo (ping) r
12	0.000076	192.168.11.1	192.168.11.3	ICMP	54 Echo (ping) r
13	0.000096	192.168.11.1	192.168.11.3	ICMP	54 Echo (ping) r
14	0.000114	192.168.11.3	192.168.11.1	ICMP	54 Echo (ping) r
15	0.000120	192.168.11.1	192.168.11.3	ICMP	54 Echo (ping) r
16	0.000138	192.168.11.3	192.168.11.1	ICMP	54 Echo (ping) r
17	0.000630	192.168.11.3	192.168.11.1	ICMP	54 Echo (ping) r
18	0.000848	192.168.11.1	192.168.11.3	ICMP	54 Echo (ping) r
19	0.010038	192.168.11.1	192.168.11.3	ICMP	54 Echo (ping) r
20	0.012043	192.168.11.3	192.168.11.1	ICMP	54 Echo (ping) r
21	0.012532	192.168.11.1	192.168.11.3	ICMP	54 Echo (ping) r
22	0.014699	192.168.11.3	192.168.11.1	ICMP	54 Echo (ping) r
23	0.021309	192.168.11.3	192.168.11.1	ICMP	54 Echo (ping) r
24	0.021702	192.168.11.1	192.168.11.3	ICMP	54 Echo (ping) r
25	0.023236	192.168.11.3	192.168.11.1	ICMP	54 Echo (ping) r
26	0.024461	192.168.11.1	192.168.11.3	ICMP	54 Echo (ping) r
27	0.025850	192.168.11.3	192.168.11.1	ICMP	54 Echo (ping) r
28	0.026374	192.168.11.1	192.168.11.3	ICMP	54 Echo (ping) r
29	0.029235	192.168.11.3	192.168.11.1	ICMP	54 Echo (ping) r
30	0.032298	192.168.11.3	192.168.11.1	ICMP	54 Echo (ping) r
31	0.032838	192.168.11.1	192.168.11.3	ICMP	54 Echo (ping) r
32	0.034538	192.168.11.3	192.168.11.1	ICMP	54 Echo (ping) r
33	0.034984	192.168.11.1	192.168.11.3	ICMP	54 Echo (ping) r
34	0.037604	192.168.11.3	192.168.11.1	ICMP	54 Echo (ping) r
35	0.042358	192.168.11.3	192.168.11.1	ICMP	54 Echo (ping) r

Identifier (LE): 512 (0x0200)
Sequence Number (BE): 22868 (0x5954)
Sequence Number (LE): 21593 (0x5459)
[\[Response frame: 2\]](#)

▼ Data (12 bytes)
Data: 6269742e6c792f777061325f

0000	b0 48 7a d2 dd 74 b8 27 eb 35 ab 17 08 00 45 00	·Hz·t·'·5···E·
0010	00 28 a6 eb 40 00 40 01 fc 94 c0 a8 0b 03 c0 a8	·(·@······
0020	0b 01 08 00 89 60 00 02 59 54 62 69 74 2e 6c 79	·····YIbit.ly
0030	2f 77 70 61 32 5f	/wpa2_

Figura 7: Análisis del archivo -dec.cap generado

De esta forma, se aprecia en cada paquete (en el campo data) la ruta desde la cual se debe descargar el archivo: **bit.ly/wpa2_**

3. Desarrollo (PASO 2)

3.1. indica script para modificar diccionario original

A lo largo del desarrollo del programa, el alumno se encontro con ciertos errores que debio corregir, como por ejemplo: *Ocurrió un error: string index out of range*, *Ocurrió un error: 'utf-8' codec can't decode byte 0xf1 in position 923: invalid continuation byte*. Finalmente se sortearon los problemas, dando como resultado el siguiente código:

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

```
1 import sys
2
3 # Función para aplicar las transformaciones a una línea
4 def transform_line(line):
5     if line and line[0].isalpha():
6         return line[0].upper() + line[1:] + "0"
7     elif line and line[0].isdigit():
8         return ""
9     else:
10        return line
11
12 # Verificar si se proporcionó el nombre del archivo en la línea de comandos
13 if len(sys.argv) != 2:
14     print("Uso: python programa.py nombre_del_archivo")
15     sys.exit(1)
16
17 # Obtener el nombre del archivo de la línea de comandos
18 nombre_archivo = sys.argv[1]
19
20 # Nombre del archivo de salida
21 nombre_archivo_salida = "rockyou_mod.dic"
22
23 lineas_procesadas = 0 # Contador de líneas procesadas
24
25 try:
26     with open(nombre_archivo, "r", encoding="ISO-8859-1") as archivo_entrada, open(nombre_archivo_salida, "w") as archivo_salida:
27
28         for linea in archivo_entrada:
29             linea_transformada = transform_line(linea.strip()) # Eliminar espacios en blanco alrededor
30             if linea_transformada:
31                 archivo_salida.write(linea_transformada + "\n")
32                 lineas_procesadas += 1
33 except FileNotFoundError:
34     print(f"El archivo '{nombre_archivo}' no fue encontrado.")
35 except Exception as e:
36     print(f"Ocurrió un error: {str(e)}")
37
38 print(f"Transformación completada. Resultado guardado en 'rockyou_mod.dic'.")
39 print(f"Líneas procesadas: {lineas_procesadas}")
```

Figura 8: Programa para modificar el diccionario original

3.2. cantidad de passwords finales que contiene rockyou_mod.dic

Al ejecutar el programa expuesto anteriormente, indicando el archivo de entrada (rockyou.txt), se obtiene el nuevo archivo con las modificaciones solicitadas. Este archivo contiene 11.059.725 líneas, como sigue:

```
(john@kali) - [~/Documentos/cripto/lab_03]
$ python3 programa.py rockyou.txt
Transformación completada. Resultado guardado en 'rockyou_mod.dic'.
Líneas procesadas: 11059725
```

Figura 9: Generación del nuevo archivo rockyou_mod.dec

4. Desarrollo (Paso 3)

4.1. obtiene contraseña con hashcat con potfile

En primer lugar se debe instalar el programa hashcat, para esto, se debe acceder a la página web oficial del programa (<https://hashcat.net/hashcat/>), desde donde se descargara la

ultima versión. Una vez descargada, se procede con la instalación:

```
(john@kali) - [~/Descargas]
$ 7z x hashcat-6.2.6.7z

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=es_CL.UTF-8,Utf16=on,HugeFiles=on,64 bits,12 CPUs Intel(R) Core(TM) i7-10750H CPU @ 2.60
GHz (A0652),ASM,AES-NI)

Scanning the drive for archives:
1 file, 20951515 bytes (20 MiB)

Extracting archive: hashcat-6.2.6.7z
--
Path = hashcat-6.2.6.7z
Type = 7z
Physical Size = 20951515
Headers Size = 19192
Method = LZMA2:384m LZMA:20 BCJ2
Solid = +
Blocks = 2
Everything is Ok

Folders: 46
Files: 2519
Size: 279323735
Compressed: 20951515
```

Figura 10: Instalación Hashcat

Para seguir con el ataque, se debe hacer uso del siguiente comando:

```
./hashcat.bin -m <modo_de_ataque><archivo_de_hashes><diccionario>
```

El archivo descargado desde bit.ly es una captura de paquetes, sin embargo, hashcat necesita un archivo de hashes. Para convertir esta captura al formato adecuado, se debe hacer uso de **cap2hccapx**, de la herramienta **hcxtools**, por lo que se procede con la instalación de la misma:

```
(john@kali) - [~/Descargas/hashcat-6.2.6]
$ sudo apt install hcxtools
Leyendo lista de paquetes ... Hecho
Creando árbol de dependencias ... Hecho
Leyendo la información de estado ... Hecho
Se instalarán los siguientes paquetes adicionales:
 hcxumptool
Se instalarán los siguientes paquetes NUEVOS:
 hcxumptool hcxtools
0 actualizados, 2 nuevos se instalarán, 0 para eliminar y 1861 no actualizados.
Se necesita descargar 250 kB de archivos.
Se utilizarán 773 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n]
Des:1 http://kali.download/kali kali-rolling/main amd64 hcxumptool amd64 6.3.1-1 [75,6 kB]
```

Figura 11: Instalación hcxtools

Luego de 2 horas intentando convertir el archivo .pcap a un archivo compatible (.hccapx), la estudiante descubre (mas bien, le dijo un amigo), que existe una pagina para convertir el archivo .pcap a uno de hashes (<https://hashcat.net/cap2hashcat/>), la que genera un archivo con formato **.hc22000**. Este archivo si es el indicado para obtener la contraseña, por lo que se procede con el uso del siguiente comando:

```
./hashcat.bin -m 22000 190271_1697599616.hc22000 rockyou_mod.dic
--potfile-path=patfile.txt --force
```

EL output en el punto consecutivo.

4.2. identifica nomenclatura del output

Tras aplicar el comando exhibido en el apartado anterior, el output obtenido es el siguiente:

```
(john@kali)-[~/Descargas/hashcat-6.2.6]
$ ./hashcat.bin -m 22000 190271_1697599616.hc22000 rockyou_mod.dic --potfile-path=patfile.txt --force
hashcat (v6.2.6) 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.

OpenCL API (OpenCL 3.0 PoCL 3.0+debian Linux, None+Asserts, RELOC, LLVM 13.0.1, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]

=====
* Device #1: pthread-Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz, 6784/13632 MB (2048 MB allocatable), 12MCU

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: 3 MB

Dictionary cache built:
* Filename..: rockyou_mod.dic
* Passwords.: 11059725
* Bytes.....: 120009733
* Keyspace..: 11059707
* Runtime...: 1 sec

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

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 22000 (WPA-PBKDF2-PMKID+EAOL)
Hash.Target.....: 190271_1697599616.hc22000
Time.Started.....: Wed Oct 18 00:41:37 2023, (1 sec)
Time.Estimated...: Wed Oct 18 00:41:38 2023, (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (rockyou_mod.dic)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 3225 H/s (9.47ms) @ Accel:128 Loops:512 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 2907/11059707 (0.03%)
Rejected.....: 1371/2907 (47.16%)
Restore.Point....: 0/11059707 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: Password0 -> Dangerous0
Hardware.Mon.#1..: Temp: 80c Util: 26%

Started: Wed Oct 18 00:40:51 2023
Stopped: Wed Oct 18 00:41:39 2023
```

Figura 12: Output hashcap - contraseña obtenida con potfile

Se aprecian distintos parametros que se utilizaron y configuraron para crackear la contraseña, dentro de los que podemos destacar los siguientes:

- **Hash Mode (Modo de Hash):** Este es el modo de hash utilizado en el ataque. En este caso, el modo de hash es 22000, que corresponde a "WPA-PBKDF2-PMKID+EAPOL". Este modo se usa comúnmente para descifrar contraseñas de redes Wi-Fi protegidas con WPA/WPA2, sin embargo en este caso fue usado para una red con cifrado WEP.
- **Hash Target (Objetivo de Hash):** Es el archivo de hash que se está intentando descifrar.
- **Time Started (Tiempo de Inicio):** Muestra la fecha y hora en la que se inició el proceso de descifrado.
- **Kernel Feature (Característica del Kernel):** Muestra información sobre el kernel utilizado en el proceso de descifrado.
- **Guess Base (Base de Suposiciones):** Indica que se está utilizando un diccionario de contraseñas para realizar los intentos de descifrado. El diccionario de contraseñas utilizado es rockyou_mod.dic".
- **Speed (Velocidad):** Muestra la velocidad a la que Hashcat está probando contraseñas. En este caso, la velocidad es de aproximadamente 20,425 hashes por segundo.
- **Recovered (Recuperado):** Indica cuántas contraseñas se han recuperado con éxito. En este caso, se recuperó una contraseña.
- **Progress (Progreso):** Muestra la progresión del proceso de descifrado. En este caso, se han realizado 5,533 intentos de descifrado, lo que representa aproximadamente el 0.05 % del total de contraseñas en el diccionario.
- **Candidate Engine (Motor de Candidatos) :**Indica cómo se generan las contraseñas candidatas. En este caso, se utiliza el "Device Generator".
- **Candidates (Candidatos):** Muestra el rango de contraseñas que se están probando en ese momento. Por ejemplo, "Password0 -¡Fuckyou690"indica que se están probando contraseñas en ese rango.
- **Hardware Monitoring (Monitoreo de Hardware):** Proporciona información sobre la temperatura y la utilización del hardware durante el proceso.
- **Started (Comenzado) y Stopped (Detenido):** Muestra la fecha y hora de inicio y finalización del proceso de descifrado.

Ahora bien, lo que nos interesa es la nomenclatura de la respuesta del ataque. La respuesta es la siguiente:

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

Así, se puede deducir que la nomenclatura es la siguiente:

Hash:Salt:PMKID:SSID

De esta forma, los parametros son:

- **Hash** : 1813acb976741b446d43369fb96dbf90
- **Salt** : b0487ad2dc18
- **PMKID** : eede678cdf8b
- **Nombre de la red**: VTR-1645213
- **Contraseña**: Security0

4.3. obtiene contraseña con hashcat sin potfile

En este caso, no se requiere hacer uso de potfile, que es un archivo en el que Hashcat registra las contraseñas que ha descifrado durante un ataque. Hashcat compara las contraseñas descifradas con las contraseñas en el potfile para evitar ataques innecesarios a contraseñas que ya se han descifrado en ataques anteriores. El comando utilizado a continuacion:

```
./hashcat.bin -m 22000 190271_1697599616.hc22000 rockyou_mod.dic  
--potfile-disable --force
```

4.4. identifica nomenclatura del output

El output generado es muy similar al del apartado anterior, solo que ahora se demora considerablemente menos.

```

(john@kali)-[~/Descargas/hashcat-6.2.6]
$ ./hashcat.bin -m 22000 190271_1697599616.hc22000 rockyou_mod.dic --potfile-disable --force
hashcat (v6.2.6) 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.

OpenCL API (OpenCL 3.0 PoCL 3.0+debian Linux, None+Asserts, RELOC, LLVM 13.0.1, SLEEP, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]

* Device #1: pthread-Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz, 6784/13632 MB (2048 MB allocatable), 12MCU

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: 3 MB

Dictionary cache hit:
* Filename..: rockyou_mod.dic
* Passwords.: 11059707
* Bytes.....: 120009733
* Keyspace...: 11059707

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

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 22000 (WPA-PBKDF2-PMKID+EAPOl)
Hash.Target.....: 190271_1697599616.hc22000
Time.Started....: Wed Oct 18 00:58:57 2023, (0 secs)
Time.Estimated...: Wed Oct 18 00:58:57 2023, (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (rockyou_mod.dic)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 20425 H/s (9.19ms) @ Accel:256 Loops:256 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 5533/11059707 (0.05%)
Rejected.....: 2461/5533 (44.48%)
Restore.Point...: 0/11059707 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: Password0 -> Fuckyou690
Hardware.Mon.#1..: Temp: 82c Util: 29%

Started: Wed Oct 18 00:58:56 2023
Stopped: Wed Oct 18 00:58:59 2023

```

Figura 13: Output hashcap - contraseña obtenida sin potfile

Se aprecia que los parametros configurados son los mismos que la vez anterior, con pequeñas variaciones, por lo que no se profundizara en ello.

La nomenclatura del output es la misma:

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

Así, se puede deducir que la nomenclatura es la siguiente:

Hash:Salt:PMKID:SSID

De esta forma, los parametros son:

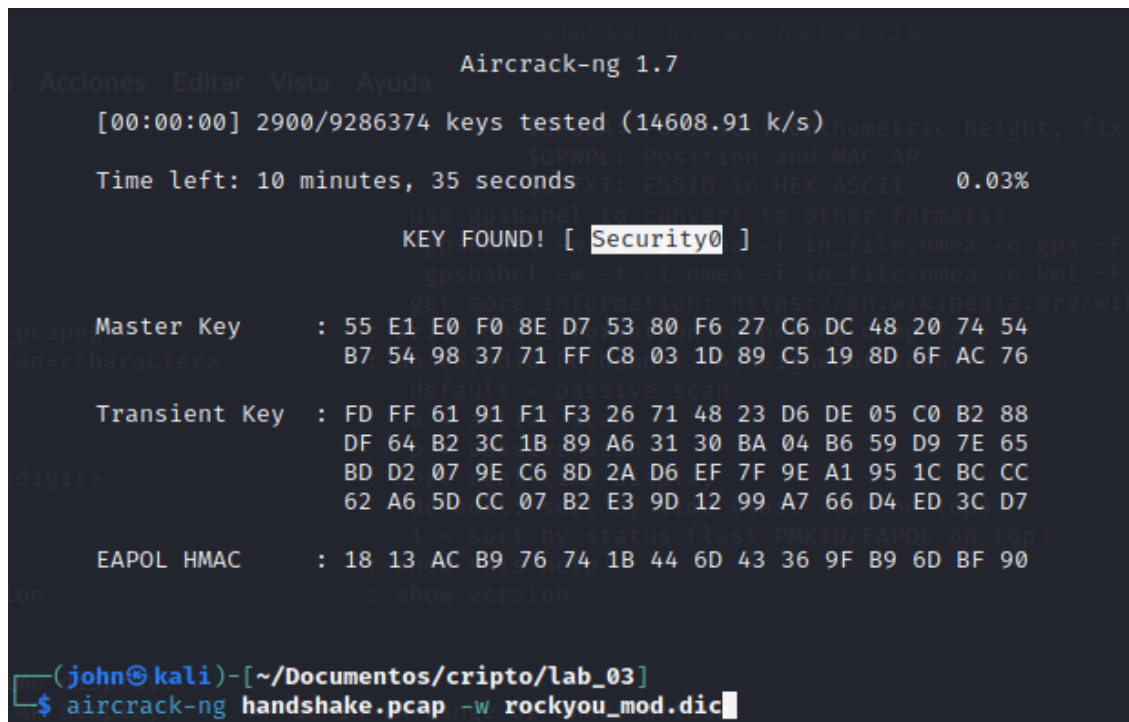
- **Hash** : 1813acb976741b446d43369fb96dbf90
- **Salt** : b0487ad2dc18
- **PMKID** : eede678cdf8b
- **Nombre de la red**: VTR-1645213
- **Contraseña**: Security0

4.5. obtiene contraseña con aircrack-ng

Para la obtencion de la contraseña con aircrack-ng, el procedimiento y comandos son ya conocidos, pues se ha trabajado con ellos anteriormente, y tambien en pasos anteriores, por lo que no se profundizara mucho al respecto. El comando utilizado es el siguiente:

```
aircrack-ng handshake.pcap -w rockyou_mod.dic
```

El output obtenido por este comando es el siguiente:



```
Aircrack-ng 1.7
Acciones: Editar, Vista, Ayuda
[00:00:00] 2900/9286374 keys tested (14608.91 k/s)
Time left: 10 minutes, 35 seconds
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 ED 3C D7

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

(john@kali)-[~/Documentos/cripto/lab_03]
$ aircrack-ng handshake.pcap -w rockyou_mod.dic
```

Figura 14: Output al utilizar aircrack-ng

Se pueden apreciar entonces los campos:

- **Diccionario:** El diccionario utilizado es el modificado a partir del descargado de rock-you. Recordar que a este diccionario se le borraron los string que comienzan con números, y la primera letra de los restantes se paso a su versión capital.
- **ssid:** Es el ssid de la red a auditar.
- **aNonce:** El ANonce es un valor aleatorio generado por el punto de acceso (AP) o autenticador. Su propósito principal es evitar ataques de repetición, ya que es diferente en cada intento de autenticación. El ANonce se utiliza para calcular la PMK (Pairwise Master Key) y, a partir de ella, se derivan las claves necesarias para la comunicación segura.

```

4 0.002402      Tp-LinkT_d2:dc:18 (... 802.11) 10 Acknowledgement, Flag
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, Flag

> Frame 5: 133 bytes on wire (1064 bits), 133 bytes captured (1064 bits)
> IEEE 802.11 QoS Data, Flags: .....F.
> Logical-Link Control
- 802.1X Authentication
  Version: 802.1X-2004 (2)
  Type: Key (3)
  Length: 95
  Key Descriptor Type: EAPOL RSN Key (2)
  [Message number: 1]
  Key Information: 0x008a
  Key Length: 16
  Replay Counter: 1
  WPA Key Nonce: 4c2fb7eca28fba45accefd3ac5e433314270e04355b6d95086031b004a31935
  Key IV: 00000000000000000000000000000000
  WPA Key RSC: 0000000000000000
  WPA Key ID: 0000000000000000
  WPA Key MIC: 00000000000000000000000000000000
  WPA Key Data Length: 0

```

Figura 16: Obtención Anonce a partir de captura de Wireshark

- **sNonce:** El SNonce es un valor aleatorio generado por la estación o el cliente que intenta conectarse a la red. Su objetivo es también evitar ataques de repetición y garantizar la singularidad de cada intento de autenticación. El SNonce se combina con el ANonce y se utiliza para generar la misma PMK, lo que permite que ambas partes, el AP y el cliente, obtengan la misma clave de sesión.

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

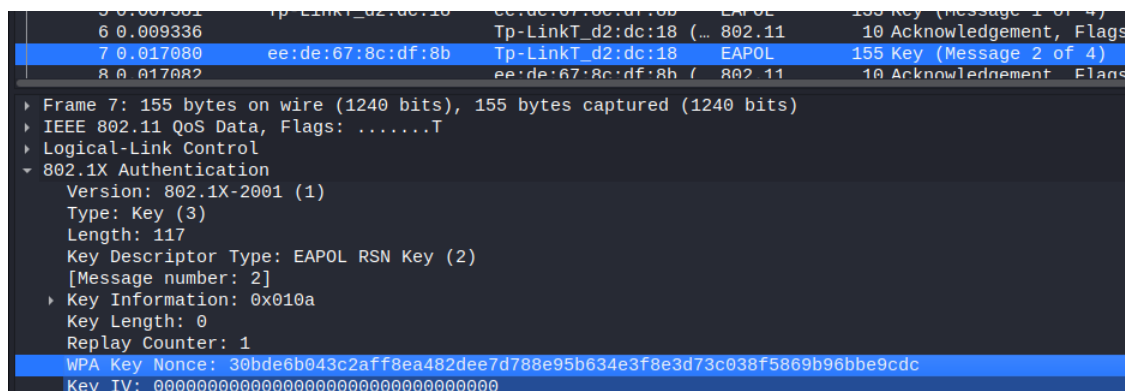


Figura 17: Obtención Snonce a partir de captura de Wireshark

- **apMac y cliMac:** Estos valores son las direcciones tanto del transmisor como del receptor, es decir, del Access Point y del dispositivo que intenta conectarse.

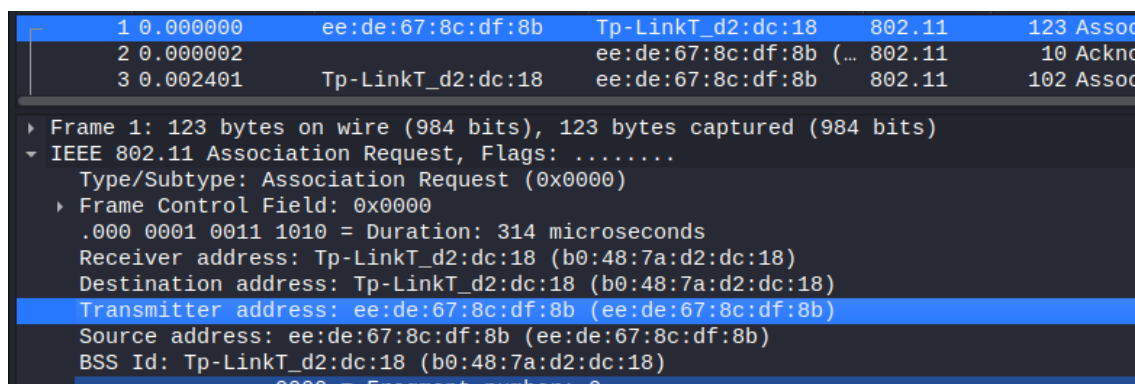


Figura 18: Obtención Mac's de origen y destino

- **MIC (Message Integrity Check) y data:** El campo MIC es un valor que representa la verificación de la integridad del mensaje. Es una suma de comprobación (hash) que se utiliza para garantizar que los datos transmitidos en el mensaje no se hayan modificado durante la transmisión. El objetivo principal es calcular y verificar el MIC para cada mensaje del handshake de cuatro vías. **Se generan MICs para los mensajes 2, 3 y 4 del handshake** y se comparan con los MICs esperados para verificar la autenticidad de los mensajes.

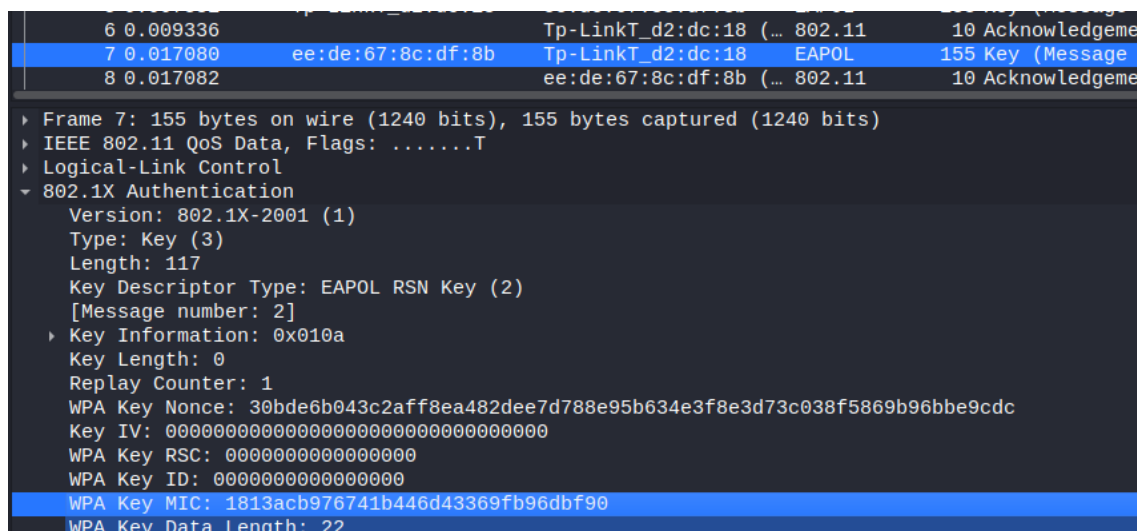


Figura 19: Obtención campo MIC

La forma de calcular los 3 MIC es la misma para cada uno de ellos. Se analizan los mensajes 2, 3 y 4 para distinguir MIC.

El campo data (2, 3 y 4) se obtiene al exportar como **Hex Stream** los campos 802.11x Authentication para los mensajes 2, 3 y 4 respectivamente.

4.7. obtiene contraseña con pycrack

Una vez configurados todos los parametros correctamente con los procedimientos anteriormente descritos, se procede con la ejecución del programa .py, como sigue:

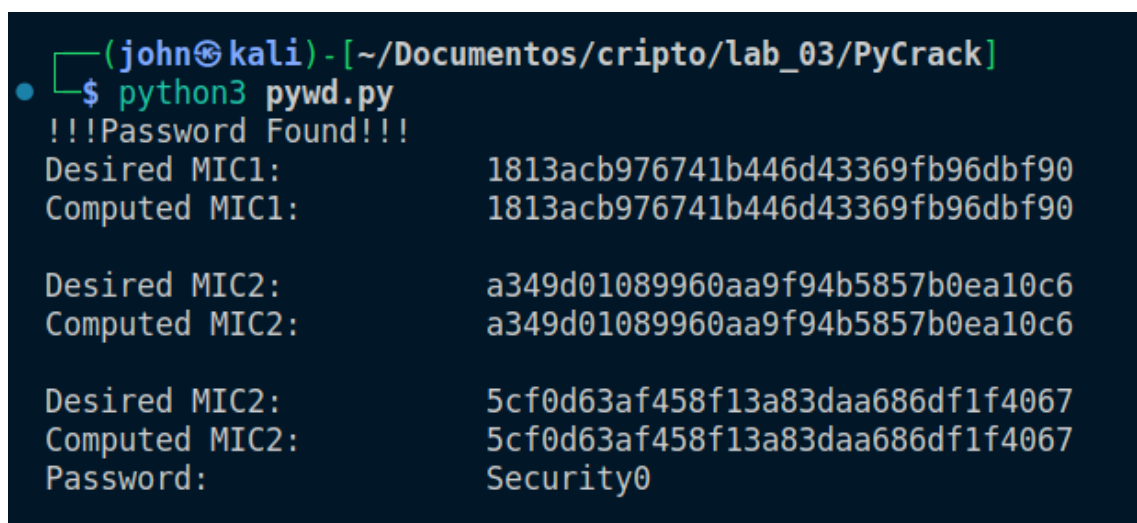


Figura 20: Obtención contraseña usando pycrack

De esta forma, se aprecia que Pycrack, al igual que las herramientas anteriores, encuentra la contraseña de la red: **Security0**.

5. Conclusiones y comentarios

En el presente documento, se realizaron diversas actividades relacionadas con la recuperación de la contraseña de una red WiFi con cifrado WEP. Para llegar a los resultados exitosos en estas actividades se usaron 3 herramientas diferentes: Hashcat, Aircrack-ng y Pycrack.

En cuanto al uso de cada herramienta, podemos distinguir los siguientes puntos relacionados con cada una de ellas:

1. **Hashcat:** es una herramienta poderosa y versátil para recuperar contraseñas mediante fuerza bruta y ataques de diccionario.
 - Puede funcionar en modo de línea de comandos y admite múltiples modos de ataque, incluido el modo de ataque WPA/WPA2, en este caso, utilizado para una red con cifrado WEP.
 - La obtención de contraseñas con Hashcat puede ser eficiente y rápida, especialmente cuando se usa una tarjeta gráfica (GPU) compatible.
 - La respuesta que entrega Hashcat es mucho mas completa y con mas información en comparación con los otros dos métodos, indicando gran variedad de campos y valores utilizados en el descifrado de la clave.
 - Si bien es cierto que Hashcat entrega una respuesta mucho mas clara y detallada que las otras herramientas, el proceso de instalación y la gran cantidad de opciones configurables que tiene pueden dificultar en un comienzo su comprensión y ejecución. De este modo, presenta una mayor dificultad la trabajar.
2. **Aircrack.ng:** Es una herramienta bien conocida para la recuperación de contraseñas de redes Wi-Fi con cifrado WEP.
 - Es específica para redes Wi-Fi y tiene características para capturar y analizar paquetes de redes inalámbricas.
 - Aircrack-ng es útil para redes WEP, pero no es adecuada para descifrar contraseñas de redes WPA/WPA2 debido a la complejidad de su cifrado.
 - Dada la dedicación de aircrack-ng solo a redes wifi, esta tiene menos parametros configurables. Esto permite que sea mas fácil trabajar con esta herramienta.
3. **Pycrack:** Es una herramienta más específica y personalizada desarrollada en Python para la recuperación de contraseñas de redes Wi-Fi con cifrado WEP.
 - Requiere una configuración más específica y personalizada en comparación con las otras herramientas.

- Pycrack ofrece un mayor control sobre los detalles de los mensajes EAPOL y otros parámetros, lo que lo hace adecuado para fines educativos y prácticos de aprendizaje.
- Pycrack no es una herramienta como tal, es un programa python, con lo que se puede apreciar paso a paso como es que se descifra la contraseña

Finalmente, en cuanto a las dificultades encontradas podemos distinguir:

- AL trabajar con Hashcat, esta herramienta recibe como parametro un archivo de hashes en vez de la captura .pcap. Esto presento un gran inconveniente, pues en principio se intento convertir este archivo al formato adecuado utilizando herramientas de *hcxdump-tool*, sin embargo, la forma mas conveniente de hacerlo era mediante la pagina.
- La comprensión de los detalles de los mensajes EAPOL y los valores necesarios para Pycrack requirieron un conocimiento profundo de los protocolos de autenticación.
- En Hashcat, la configuración de parámetros y modos de ataque específicos puede ser complicada, especialmente para usuarios principiantes.

Finalmente, se concluye que cada herramienta tiene sus ventajas y desventajas, y la elección de cuál utilizar depende de la situación específica. Hashcat es una opción versátil para una variedad de escenarios, Aircrack-ng es excelente para redes WEP y Pycrack ofrece una experiencia más personalizada y educativa.