

# Informe Laboratorio 3

## Sección 1

Alumno: Kerssen Barros  
e-mail: kerssen.barros@mail.udp.cl

Mayo de 2024

## Índice

<b>1. Descripción de actividades</b>	<b>2</b>
<b>2. Desarrollo (PASO 1)</b>	<b>3</b>
2.1. 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 . . . . .	4
2.4. Indica el tiempo que demoró en obtener la password . . . . .	5
2.5. Descifra el contenido capturado . . . . .	5
2.6. Describe como obtiene la url de donde descargar el archivo . . . . .	6
<b>3. Desarrollo (PASO 2)</b>	<b>7</b>
3.1. Script para modificar diccionario original . . . . .	7
3.2. Cantidad de passwords finales que contiene rockyou_mod.dic . . . . .	8
<b>4. Desarrollo (Paso 3)</b>	<b>8</b>
4.1. Obtiene contraseña con hashcat con potfile . . . . .	8
4.2. Nomenclatura del output . . . . .	10
4.3. Obtiene contraseña con hashcat sin potfile . . . . .	11
4.4. Nomenclatura del output . . . . .	11
4.5. Obtiene contraseña con aircrack-ng . . . . .	12
4.6. Identifica y modifica parámetros solicitados por pycrack . . . . .	13
4.7. Obtiene contraseña con pycrack . . . . .	19
<b>5. Conclusiones y comentarios</b>	<b>20</b>
5.1. Problemáticas en el Laboratorio . . . . .	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 Rockyou (utilizado ampliamente en el mundo del pentesting). Haga un script que para cada string contenido en el diccionario, reemplace la primera letra por su letra en capital y agregue un cero al final de la password.

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

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

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

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

## 2. Desarrollo (PASO 1)

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

Al analizar todas las redes visibles en el laboratorio, se puede apreciar que la red del informante es la única que posee un sistema de cifrado **WEP**, haciéndola reconocible entre todas las otras redes mostradas.

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC CIPHER	AUTH	ESSID
E4:AB:89:E4:96:71	-1	0	0	0	11	-1		<length: 0>
C6:BC:FB:43:F2:E8	-1	0	0	0	6	-1		<length: 0>
B0:48:7A:D2:DD:74	-58	174	2744	28	6	54e WEP WEP		WEP
82:C3:31:4C:91:31	-69	67	7	0	11	180 WPA2 CCMP	PSK	Alexis
B0:1F:8C:E1:B2:03	-80	18	0	0	1	130 OPN		Alumnos-UDP

Figura 1: Red del informante.

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC CIPHER	AUTH	ESSID
00:28:97:28:40:00	-1	0	0	0	11	-1		<length: 0>
FA:61:60:E8:2F:82	-88	0	0	0	6	-1		<length: 0>
10:F0:68:D9:B6:A9	-73	0	0	0	6	130 WPA2 CCMP	PSK	Eventos
20:F3:75:B5:60:A5	-79	2	2	0	1	540 WPA2 CCMP	PSK	MY HOME
C0:56:27:4F:AE:86	-76	3	0	0	11	130 WPA2 CCMP	PSK	HALCAYAGA
FB:20:A9:81:14:88	-87	4	0	0	11	270 WPA2 CCMP	PSK	Santiago
14:51:20:57:6C:E8	-89	6	1	0	6	270 WPA2 CCMP	PSK	DASF984
B0:1F:8C:E0:E8:83	-75	1	0	0	1	130 OPN		Alumnos-UDP
00:25:00:FF:94:43	-1	0	0	0	6	-1	OPN	<length: 0>
00:10:CC:03:12:00	-1	0	0	0	11	-1		<length: 0>
B0:48:7A:D2:DD:74	-52	521	8589	0	6	54e WEP WEP		WEP
58:EF:68:47:59:C8	-69	374	0	0	6	130 OPN		cableadatelematica-invitado
B0:1F:8C:E2:14:A2	-72	124	0	0	11	130 WPA3 CCMP	OWE	<length: 0>
82:C3:31:4C:91:31	-66	183	21	0	11	180 WPA2 CCMP	PSK	Alexis
AC:F8:CC:40:0A:D0	-80	20	0	0	11	130 WPA2 CCMP	PSK	VTR-9510041
E4:AB:89:08:B3:78	-72	10	0	0	1	130 WPA2 CCMP	PSK	Manval
90:FC:11:86:B6:89	-68	153	1267	3	11	130 WPA2 CCMP	PSK	Telematica
50:EF:68:47:59:C6	-83	221	0	0	6	130 WPA2 CCMP	PSK	cableadatelematica
84:D8:1B:C6:83:E9	-72	178	0	0	2	195 WPA2 CCMP	PSK	FAMILIAGL_EXT
44:48:B9:40:AD:F8	-84	5	0	0	11	130 WPA2 CCMP	PSK	LASR
2C:98:82:A2:82:80	-78	133	0	0	6	720 WPA2 CCMP	PSK	Depto987
B0:1F:8C:E2:14:A3	-72	121	0	0	11	130 OPN		Alumnos-UDP
CC:ED:DC:1C:0E:71	-78	143	0	0	8	130 WPA2 CCMP	PSK	Jpablov
B0:1F:8C:E2:14:A4	-72	113	14	0	11	130 WPA3 CCMP	OWE	<length: 0>
B0:1F:8C:E2:14:A1	-63	136	0	0	11	130 OPN		Invitados-UDP
B0:1F:8C:E2:14:A0	-75	128	0	0	11	130 WPA3 CCMP	SAE	Sala Hibrida-UDP
B0:1F:8C:E2:14:A7	-62	121	0	0	11	130 WPA2 CCMP	MGT	Administrativos-UDP
B0:1F:8C:E2:14:A5	-71	111	0	0	11	130 OPN		VIP-UDP
B0:1F:8C:E2:14:A6	-73	113	0	0	11	130 WPA3 CCMP	OWE	<length: 0>
B0:1F:8C:E1:B2:07	-77	48	0	0	1	130 WPA2 CCMP	MGT	Administrativos-UDP
18:35:01:90:C7:99	-77	3	11	0	11	130 WPA2 CCMP	PSK	VTR-6733269
AC:F8:CC:10:60:60	-82	58	19	0	1	130 WPA2 CCMP	PSK	VTR-8492879
B0:1F:8C:E1:B2:00	-77	55	0	0	1	130 WPA3 CCMP	SAE	Sala Hibrida-UDP
B0:1F:8C:E1:B2:04	-77	65	0	0	1	130 WPA3 CCMP	OWE	<length: 0>
B0:1F:8C:E1:B2:05	-76	69	0	0	1	130 OPN		VIP-UDP
B0:1F:8C:E1:B2:06	-67	57	0	0	1	130 WPA3 CCMP	OWE	<length: 0>
B0:1F:8C:E1:B2:02	-77	51	0	0	1	130 WPA3 CCMP	OWE	<length: 0>
B0:1F:8C:E1:B2:03	-77	60	0	0	1	130 OPN		Alumnos-UDP
E4:AB:89:07:57:38	-84	28	0	0	1	130 WPA2 CCMP	PSK	Soñías22 2,4G
20:F3:75:97:CE:05	-80	32	7	0	1	540 WPA2 CCMP	PSK	Russalie
44:48:B9:41:A2:D8	-78	106	0	0	1	130 WPA2 CCMP	PSK	CECI
FA:8F:CA:50:A8:EF	-86	53	0	0	1	65 OPN		Dormitorio grande
40:80:10:CD:FC:19	-81	20	0	0	6	130 WPA2 CCMP	PSK	VTR-6141955
14:CC:20:18:E8:35	-84	107	5	0	8	270 WPA2 CCMP	PSK	Jpablov EXT
8A:D8:1B:C6:83:E9	-68	198	0	0	2	195 WPA2 CCMP	PSK	<length: 0>
48:D3:43:B7:0C:61	-70	16	0	0	1	130 WPA2 CCMP	PSK	VTR-9108176-24
A0:04:8F:D1:56:88	-82	8	1	0	11	130 WPA2 CCMP	PSK	Casaplina
B0:1F:8C:E1:B2:01	-76	37	0	0	1	130 OPN		Invitados-UDP

Figura 2: Redes captadas en el laboratorio.

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

Como se observó en el ítem anterior, la red del informante está cifrada utilizando **WEP**.

Para este cifrado en particular, el motivo por el cual se necesitan al menos 5000 paquetes para descubrir la clave, se debe a la naturaleza del algoritmo y a las vulnerabilidades de este en su diseño.

Las claves *WEP* (40 o 104 bits) se combinan con el *IV* (24 bits) para formar una clave de cifrado RC4 con una longitud total de 64 o 128 bits.

Debido a la longitud limitada del *IV*, hay un número finito de *IV* posibles ( $2^{24} = 16777216$ ), por lo que estos se terminarán repitiendo en una red ocupada.

El requerimiento de capturar al menos 5000 paquetes para descifrar la clave con *aircrack* se debe a la necesidad de tener suficientes *IVs* y datos para aumentar la probabilidad de identificar correlaciones y patrones que revelen la clave.

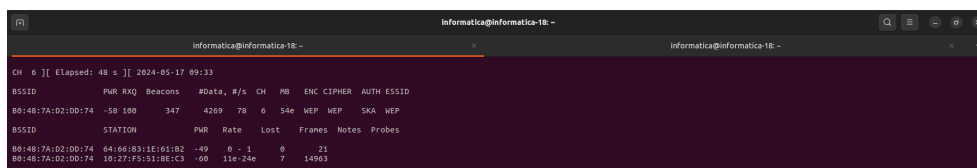
## 2.3. Obtiene la password con ataque por defecto de aircrack-ng

Para la obtención de la contraseña utilizando *aircrack-ng*, primero se realizó una captura de los paquetes con *airodump* y guardarlos en un archivo *.cap*.

```
informatica@informatica-18:~$ sudo airodump-ng -c 6 --bssid 88:4B:7A:D2:00:74 -w output-captura wlan466b31d7701
09:32:23 Created capture file "output-captura-01.cap".
```

Figura 3: Comando para capturar paquetes con *airodump*.

Donde se utilizaron las flags *-c* para especificar el canal a escuchar, *--bssid* para sólo escuchar los paquetes que provengan de la MAC especificada y *-w* para guardar los paquetes capturados en un archivo *.cap*.

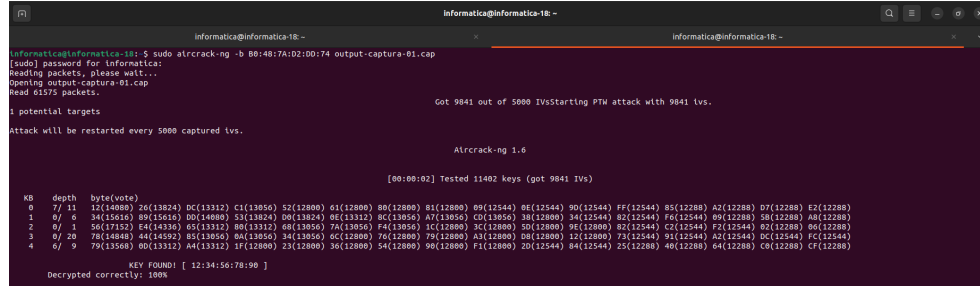


BSSID	PWR	RXQ	Beacons	#Data	#/s	CH	MB	ENC	CIPHER	AUTH	ESSID
88:4B:7A:D2:00:74	-58	100	347	4269	78	6	54e	WEP	WEP	SKA	WEP
BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes				
88:4B:7A:D2:00:74	64:66:B3:1E:01:02	-49	0 - 1	0	21						
88:4B:7A:D2:00:74	18:27:FS:31:86:13	-68	11e-24e	7	14903						

Figura 4: Captura de paquetes con *airodump*.

Mientras se seguían capturando paquetes en segundo plano, se ejecutó el comando *aircrack-ng* para analizar los paquetes capturados y crackear la clave.

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



```
Informatica@informatica-18: ~$ sudo aircrack-ng -b 00:14:87:7A:02:00:174 output-captura-01.cap
[sudo] password for informatica:
Reading packets, please wait...
Opening output-captura-01.cap
Read 61575 packets.
Got 9841 out of 5000 IVsStarting PTW attack with 9841 ivs.
1 potential targets
Attack will be restarted every 5000 captured ivs.

Aircrack-ng 1.6

[00:00:02] Tested 11402 keys (got 9841 IVs)

KB  depth  byte(vote)
0  7/ 11  12(14088) 26(13824) DC(13312) C1(13056) 52(12800) 61(12800) 80(12800) 81(12800) 09(12544) 0E(12544) 9D(12544) FF(12544) 85(12288) A2(12288) D7(12288) E2(12288)
1  0/ 0  34(15616) 80(15616) 00(14080) 53(13824) 06(13824) 0E(13312) 8C(13056) A7(13056) C0(12800) 38(12800) 34(12544) 82(12544) F6(12544) 09(12288) 98(12288) A8(12288)
2  0/ 1  56(17152) E4(14336) 65(13312) 00(13312) 08(13056) 7A(13056) F4(13056) 1C(12800) 3C(12800) 5D(12800) 9E(12800) 82(12544) C2(12544) F2(12544) 02(12288) 06(12288)
3  0/ 20  78(14848) 44(14592) 85(13056) 0A(13056) 34(13056) 0C(12800) 76(12800) 79(12800) A3(12800) D8(12800) 12(12800) 73(12544) 91(12544) A2(12544) DC(12544) FC(12544)
4  0/ 9  76(13504) 9D(13312) A4(13312) 1F(12800) 23(12800) 36(12800) 54(12800) 96(12800) F1(12800) 2D(12544) 84(12544) 25(12288) 40(12288) 64(12288) C0(12288) CF(12288)

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

Figura 5: Crackeo de clave con *aircrack*.

Donde la flag `-b` corresponde a la MAC en donde se está haciendo el ataque y *output-captura-01* el archivo que contiene los paquetes.

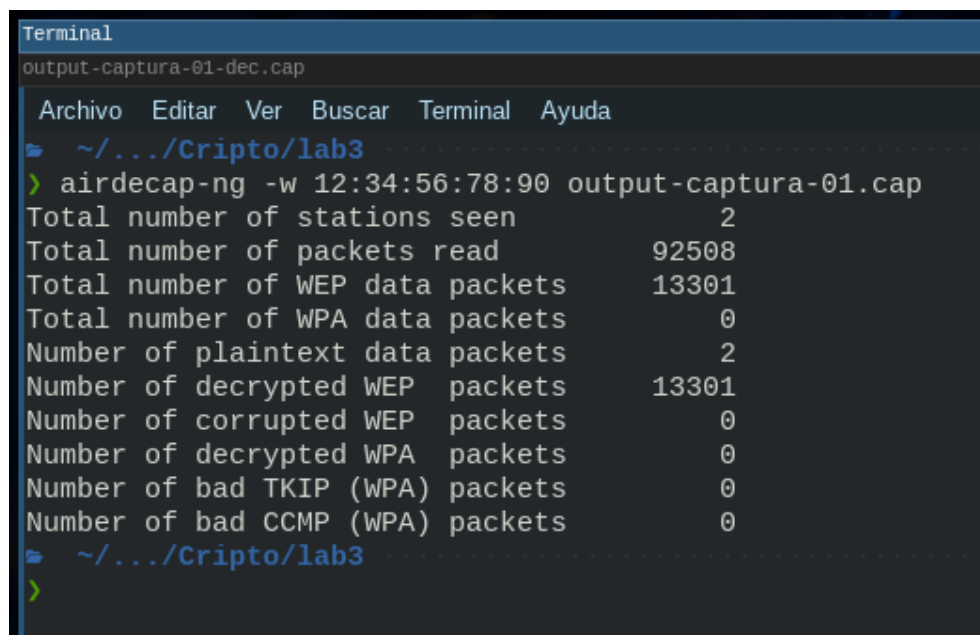
Con lo anterior, se descubre que la clave es **12:34:56:78:90**.

## 2.4. Indica el tiempo que demoró en obtener la password

El tiempo que se demoró en obtener la clave fue de **2 segundos**, como se evidenció en 2.3.

## 2.5. Descifra el contenido capturado

Una vez obtenida la clave, se hace uso del comando *airdecap* para descifrar los paquetes capturados en 2.3.



```
Terminal
output-captura-01-dec.cap

Archivo  Editar  Ver  Buscar  Terminal  Ayuda
~/.../Cripto/lab3
> airdecap-ng -w 12:34:56:78:90 output-captura-01.cap
Total number of stations seen          2
Total number of packets read          92508
Total number of WEP data packets      13301
Total number of WPA data packets      0
Number of plaintext data packets      2
Number of decrypted WEP packets      13301
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
~/.../Cripto/lab3
>
```

Figura 6: Uso de *airdecap* para descifrar los paquetes.

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

Donde la flag `-w` corresponde a la clave **WEP** para descifrar y `output-captura-01` el archivo que contiene los paquetes, guardando como resultado el archivo `output-captura-01dec.cap`.

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

Tras revisar los paquetes del archivo resultante de 2.5, se puede apreciar como en la **Data** hay una **url** hacia una página web.

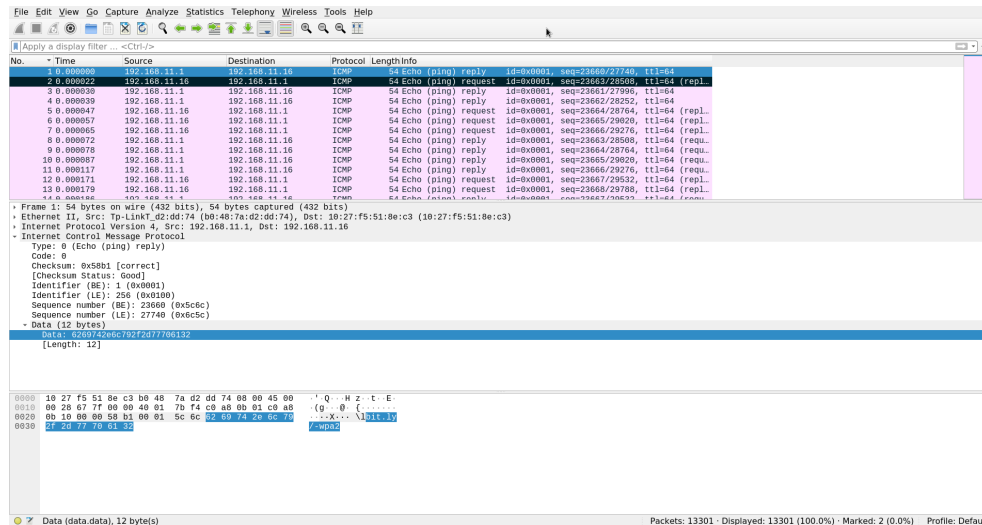


Figura 7: Url de la página encontrada en los paquetes utilizando Wireshark.

La **url** encontrada redirecciona a la página de **cloudshark.org**, en donde se encuentra el archivo **handshake.cap** a descargar.

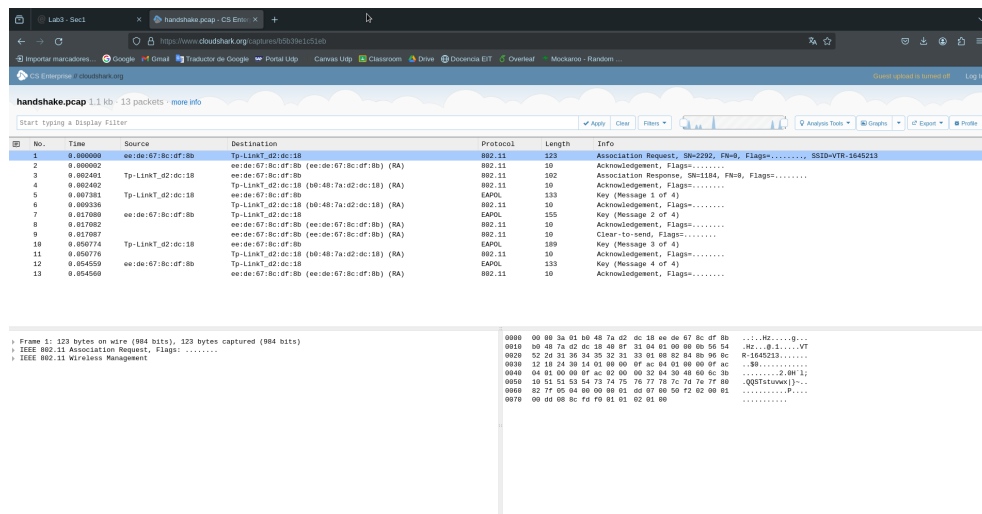


Figura 8: Página donde se encuentra el archivo a descargar.

### 3. Desarrollo (PASO 2)

#### 3.1. Script para modificar diccionario original

Para modificar el diccionario **rockyou.dic** proporcionado en el laboratorio, se utilizó el siguiente script de Python:

```
1 def modify_string(string):
2     if not string:
3         return "0"
4
5     # Cambiar la primera letra por May scula y Agregar 0 al final.
6     new_string = string[0].upper() + string[1:]
7     new_string += '0'
8
9     return new_string
10
11 def process_file(input_file, output_file):
12     with open(input_file, 'r', encoding='latin-1') as infile:
13         lines = infile.readlines()
14
15         formatted_lines = []
16         for line in lines:
17             stripped_line = line.strip()
18
19             # Omitir las l neas que est n vac as o comienzan con un n mero
20             .
21             if not stripped_line or stripped_line[0].isdigit():
22                 continue
23
24             formatted_lines.append(modify_string(stripped_line))
25             lines_count = len(formatted_lines)
26
27         with open(output_file, 'w', encoding='latin-1') as outfile:
28             for line in formatted_lines:
29                 outfile.write(line + '\n')
30
31         print(f"Se ha combiado el .dic correctamente. Cantidad de contrase as
32         : " + str(lines_count) + ".")
33
34 input_file = "rockyou.dic"
35 output_file = "rockyou_mod.dic"
36 process_file(input_file, output_file)
```

### 3.2 Cantidad de passwords finales que contiene rockyou\_mod4didDESARROLLO (PASO 3)

## 3.2. Cantidad de passwords finales que contiene rockyou\_mod.dic

Al ejecutar el script se obtiene lo siguiente:

```
~/.Cripto/lab3
> python3 alterar_dic.py
Se ha cambiado el .dic correctamente. Cantidad de contraseñas: 11059725.
~/.Cripto/lab3
>
```

Figura 9: Ejecución del script para alterar el diccionario.

En donde se puede ver que el nuevo diccionario **rockyou\_mod.dic** contiene **11059725** claves.

## 4. Desarrollo (Paso 3)

### 4.1. Obtiene contraseña con hashcat con potfile

Antes de utilizar **hashcat** para obtener la contraseña, se hace uso de la herramienta *hcxpcapngtool* (versión web) para convertir los paquetes del **handshake.cap** a formato **.hc22000**, permitiendo que las herramientas de cracking de contraseñas puedan utilizar los datos capturados de manera más eficiente.

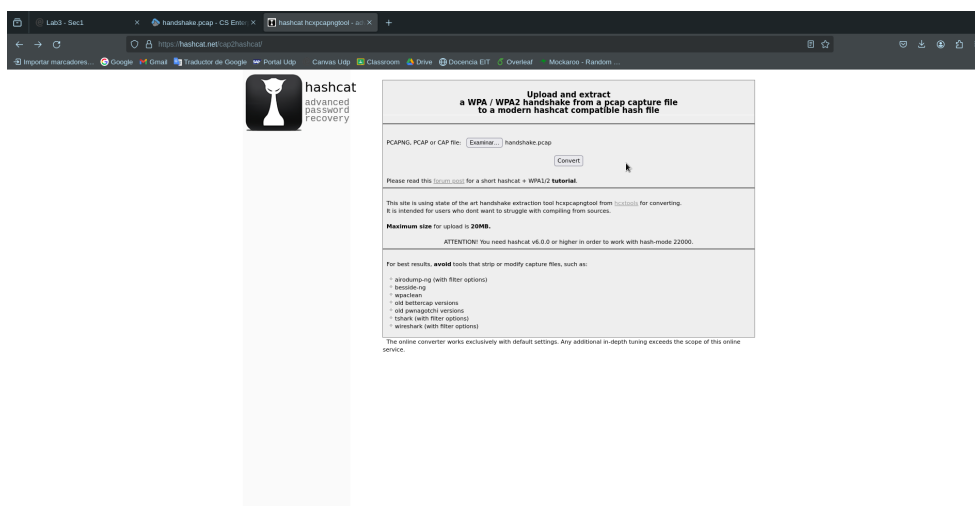


Figura 10: Página para convertir el archivo descargado a .hc22000.



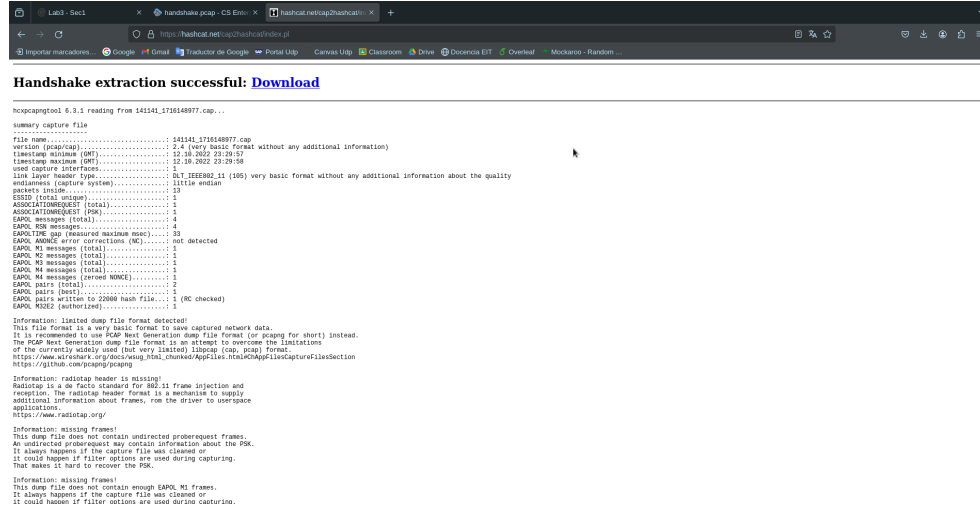


Figura 11: Archivo ya convertido.

Ya con el archivo descargado (se nombró **hash\_pcap.hc22000**), se crea el **potfile** a utilizar con el siguiente comando:

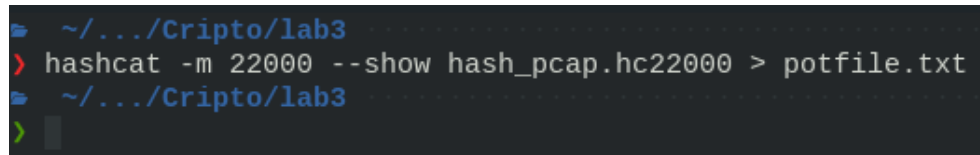


Figura 12: Comando para crear el Potfile.

Una vez creado el potfile, se procede a hacer el ataque con **hashcat**:

```

Active Editor View Source Terminal Apple
hashcat -m 22800 -a 0 hash_hcap_hc22800 rockyou_mod.dic --potfile-path potfile.txt --force
hashcat (v6.2.6-851-gb716447df) starting

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

OpenCL API (OpenCL 1.2 pocl 1.4, None+Asserts, LLVM 9.0.1, RELoc, SLEEP, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
=====
* Device #1: pthread-Intel(R) Core(TM) i5-10210U CPU @ 2.10GHz, 2783/5631 MB (1824 MB allocatable), 0MCU

Maximum 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

Hashing: Temperature abort trigger set to 90c
Host memory required for this attack: 1 MB

Dictionary cache built:
* Filename: rockyou_mod.dic
* Passwords: 11899729
* Bytes: 118974888
* Keyspace: 11899718
* Runtime: 0 secs

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

Session:..... hashcat
Status:..... Cracked
Hash Mode:..... 22800 (MD5-PBKDF2-PWID+EAPOD)
Hash Target:..... hash_hcap_hc22800
Time Started:..... Mon May 20 23:41:08 2024, (2 secs)
Time Elapsed:..... Mon May 20 23:41:09 2024, (8 secs)
Kernel Feature:..... Pure Kernel
Guess Base:..... File (rockyou_mod.dic)
Guess Queue:..... 1/1 (100.00%)
Speed #1:..... 2574 M/s (0.19s) @ Accel:256 Loops:250 Thr:1 Vec:8
Recovered:..... 1/1 (100.00%) Digits (total): 1/1 (100.00%) Digits (new)
Progress:..... 2617/11899718 (0.02%)
Rejected:..... 1169/2617 (46.20%)
Restore Point:..... 1065/11899718 (9.02%)
Restore Sub-#1:..... Salt @ Amplifier 0-1 Iteration 0-1
Candidate Engine: Device generator
Candidates #1:..... Naganadakob -> PASSWORD10
Candidates Non-#1: Temp: 72c 911: 25%
Started: Mon May 20 23:40:18 2024
Stopped: Mon May 20 23:41:09 2024

```

Figura 13: Uso de hashcat con potfile.

Donde la flag **-m** indica el modo de ataque que se va a utilizar, **-a 0** para indicar que es un ataque con diccionario, **--potfile-path** para indicar la ruta del potfile con los hashes ya crackeados y **-force** para ignorar ciertas advertencias y potenciales errores críticos.

Esto da como resultado que la contraseña es **Security0**.

## 4.2. Nomenclatura del output

**1813acb976741b446d43369fb96dbf90** es el hash MIC calculado durante el handshake.

**b0487ad2dc18** es la dirección MAC del punto de acceso inalámbrico con el que el cliente está tratando de autenticarse.

**eede678cdf8b** es la dirección MAC del cliente que está tratando de conectarse a la red.

**VTR-1645213** es el SSID de la red inalámbrica a la que el cliente está intentando conectarse.

**Security0** es la contraseña encontrada por hashcat.

### 4.3. Obtiene contraseña con hashcat sin potfile

Para obtener la contraseña sin **potfile**, se utiliza el siguiente comando de hashcat:

```

Active Editor View Docker Terminal Ayuda
hashcat [v6.2.0-851-g716447d] starting
hashcat [v6.2.0-851-g716447d] 1813acb976741b446d43369fb96dbf90 23:45:20 hashcat -m 22800 -r 0 hash_pcap.nc22800 rockyou_mod.dic --potfile-disable --force
You have enabled --force to bypass dangerous warnings and errors.
This can cause serious problems and should only be done when debugging.
Do not report hashcat issues encountered when using --force.
OpenCL API (OpenCL 1.2 pocl 1.4, None+Asserts, LLVM 9.0.1, RELOC, SLEEP, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
=====
* Device #1 pthread-Intel(R) Core(TM) i3-10110U CPU @ 2.10GHz, 2783/5631 MB (1624 MB allocatable), 4HCU
Minimum password length supported by kernel: 8
Maximum password length supported by kernel: 63
Hashes: 1 digests, 1 unique digests, 1 unique salt
Saltmap: 10 hits, 12038 entries, 888888888 max, 262144 bytes, 5/13 rotates
Males: 1
Optimizers applied:
* Zero-Byte
* Single-Hash
* Single-Salt
* Slow-Hash-100-LOOP
Watchdog: Temperature abort trigger set to 90c
Host memory required for this attack: 1 MB
Dictionary cache hit:
* Filename..: rockyou_mod.dic
* Passwords.: 11859718
* Bytes.....: 118978888
* Keyspaces..: 11859718
1813acb976741b446d43369fb96dbf90:b0487ad2dc18:eede678cdf8b-VTR-1645213:Security0
Session.....: hashcat
Status.....: Cracked
Hash Mode.....: 22800 (MD4:PMDF2-PWKID+EAPOI)
Hash Target.....: hash_pcap.nc22800
Time Started.....: Mon May 20 23:45:47 2024, (1 sec)
Time Estimated.....: Mon May 20 23:45:48 2024, (9 sec)
Kernel Feature.....: Pure Kernel
Guess Base.....: File (rockyou_mod.dic)
Guess Queue.....: 1/1 (100.00%)
Speed #1.....: 2258 MB/s (0.00ms) @ Accel:102 loops:64 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 3817/11859718 (0.03%)
Rejected.....: 1788/2857 (66.25%)
Restore Point.....: 0/11859718 (0.00%)
Restore Sub #1.....: Salt+9 Amplifier=1 Iteration=0-1
Candidate Engine.....: Device Generator
Candidates #1.....: Password -> P@SSW0RD19
Hardware #1.....: Temp: 63c still: 0ms
Started: Mon May 20 23:45:48 2024
Stopped: Mon May 20 23:45:49 2024
[ ]

```

Figura 14: Uso de hashcat sin potfile.

En donde se utilizan las mismas flags que en 4.1, cambiando la de `--potfile-path` por `--potfile-disable` para no utilizar el **potfile**.

Con lo anterior, se obtiene que la contraseña encontrada por **hashcat sin potfile** es **Security0**

### 4.4. Nomenclatura del output

**1813acb976741b446d43369fb96dbf90** es el hash MIC calculado durante el handshake.

**b0487ad2dc18** es la dirección MAC del punto de acceso inalámbrico con el que el cliente está tratando de autenticarse.

**eede678cdf8b** es la dirección MAC del cliente que está tratando de conectarse a la red.

**VTR-1645213** es el SSID de la red inalámbrica a la que el cliente está intentando conectarse.

**Security0** es la contraseña encontrada por hashcat.

## 4.5. Obtiene contraseña con aircrack-ng

Se utilizó el siguiente comando para obtener la contraseña con **aircrack**:

```

~/.ssh/Cripto/lab3
> aircrack-ng -w rockyou_mod.dic handshake.pcap
Reading packets, please wait...
Opening handshake.pcap
Read 13 packets.

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

Choosing first network as target.
Reading packets, please wait...
Opening handshake.pcap
Read 13 packets.

1 potential targets

```

Figura 15: Uso de aircrack.

En donde la flag `-w` especifica el diccionario a utilizar para el ataque y *handshake.pcap* el archivo que contiene los paquetes a atacar.

```

Aircrack-ng 1.6

[00:00:01] 3247/9296309 keys tested (5472.91 k/s)

Time left: 28 minutes, 18 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 16: Resultado de aircrack.

Tras finalizar la ejecución, se obtiene la contraseña **Security0**.

#### 4.6. Identifica y modifica parámetros solicitados por pycrack

El script **pywd.py** que ejecuta **Pycrack** pide modificar los siguientes parámetros:

[illegible]

A continuación se mostrará como se obtuvieron todos los parámetros mostrados en la imagen anterior, analizando el archivo **handshake.pcap** utilizando *Wireshark*:

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

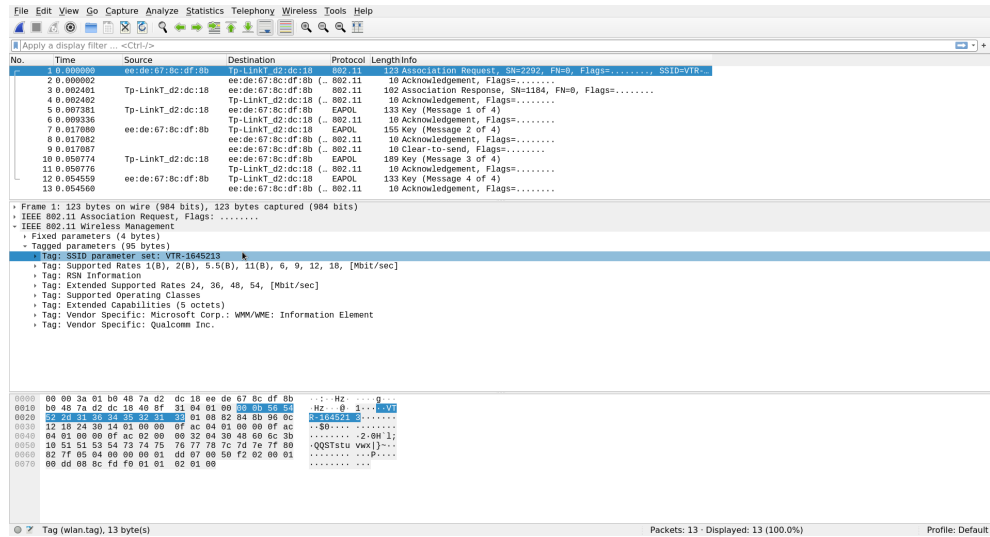


Figura 17: SSID pedido por Pycrack.

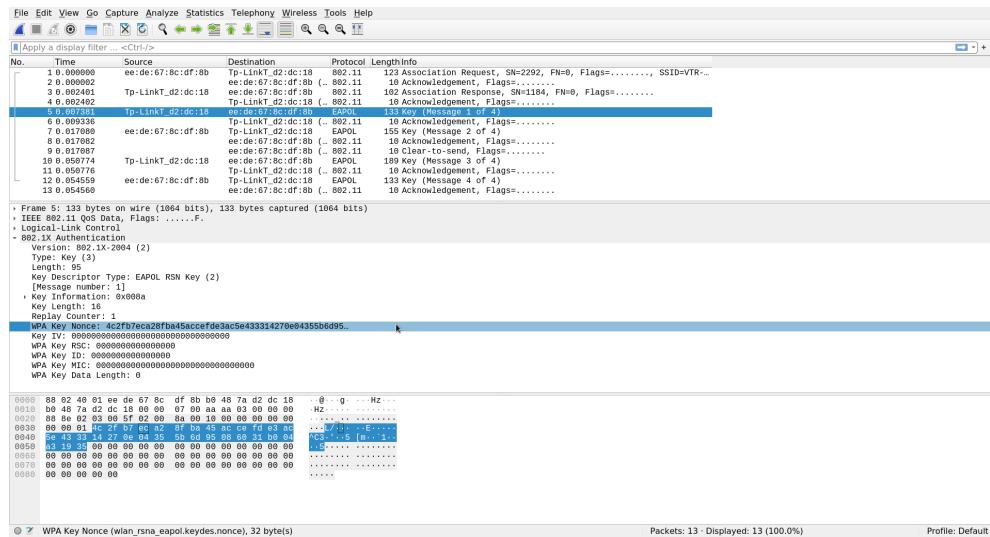


Figura 18: ANonce pedido por Pycrack.

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

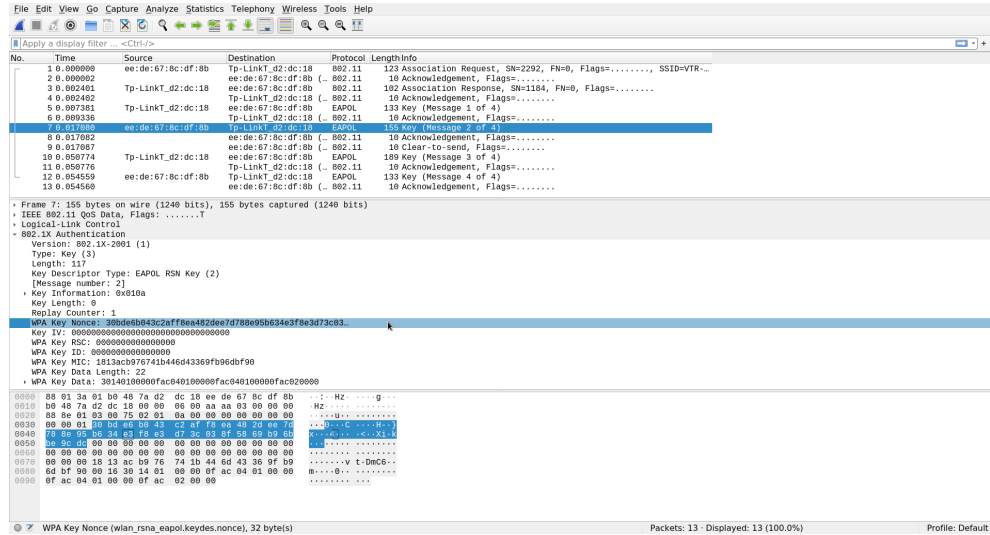


Figura 19: SNonce pedido por Pycrack.

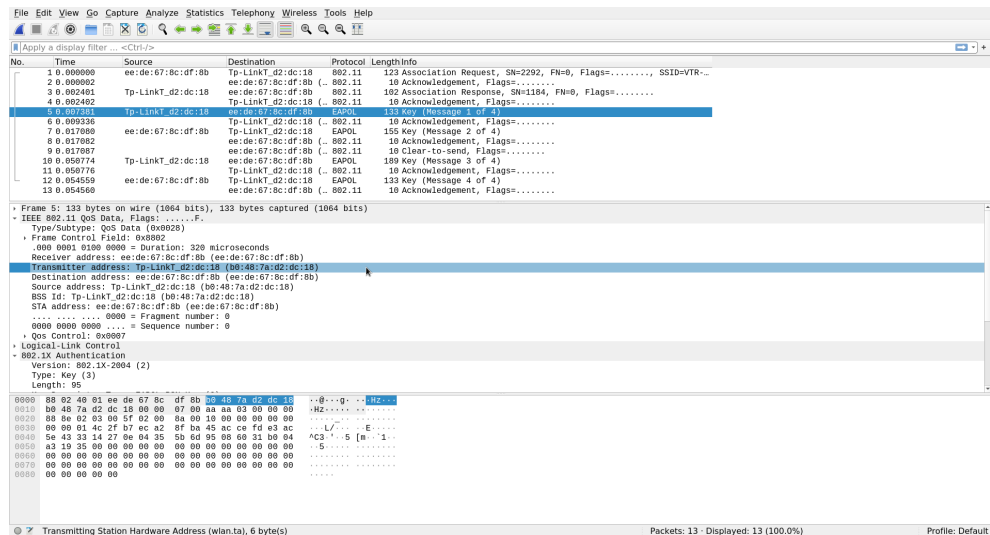


Figura 20: APMAC pedido por Pycrack.

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

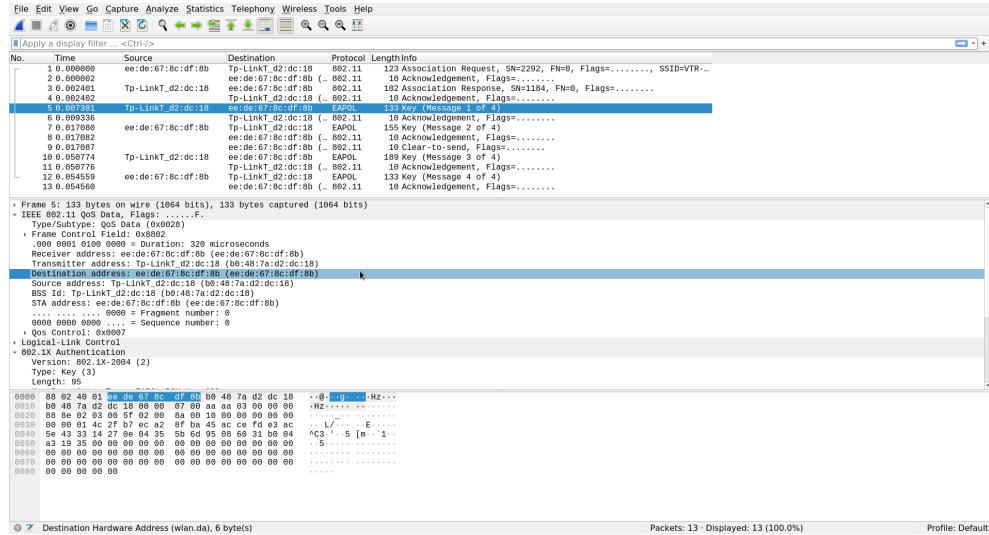


Figura 21: CLIMAC pedido por Pycrack.

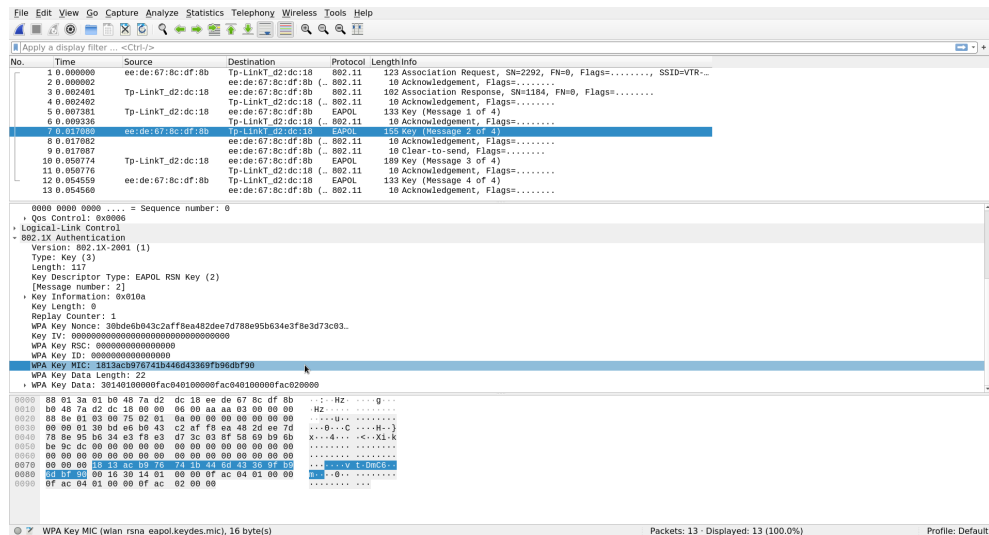


Figura 22: MIC1 pedido por Pycrack.



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

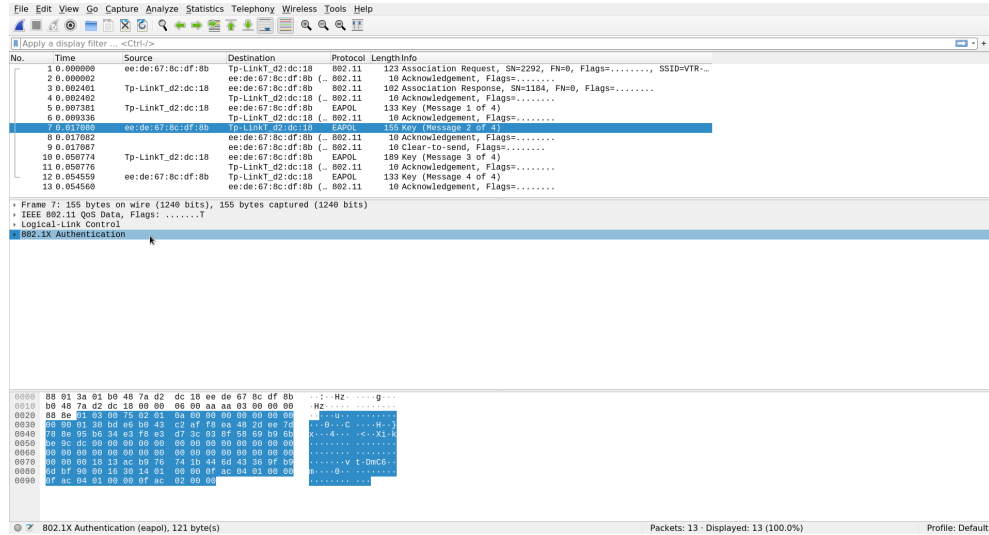


Figura 23: Data1 pedido por Pycrack.

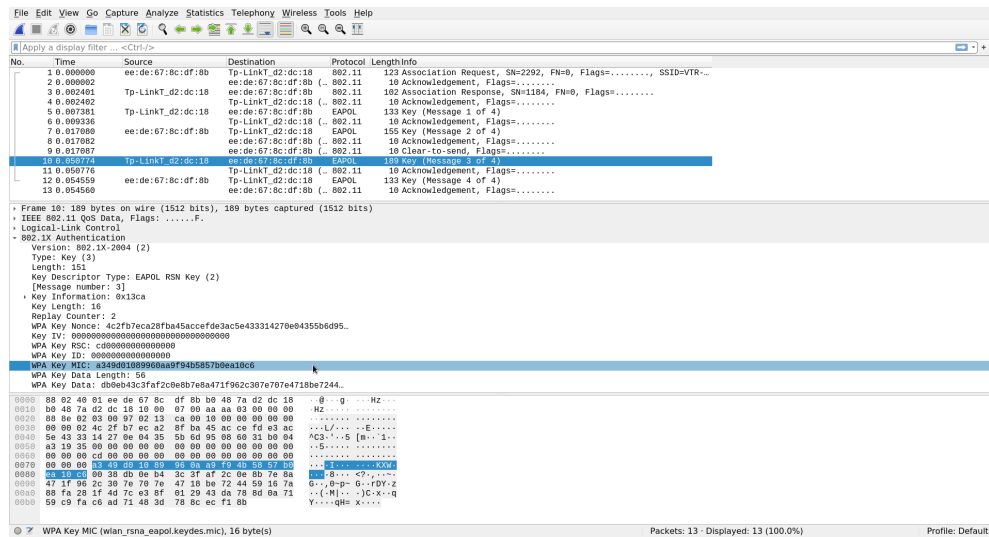


Figura 24: MIC2 pedido por Pycrack.

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

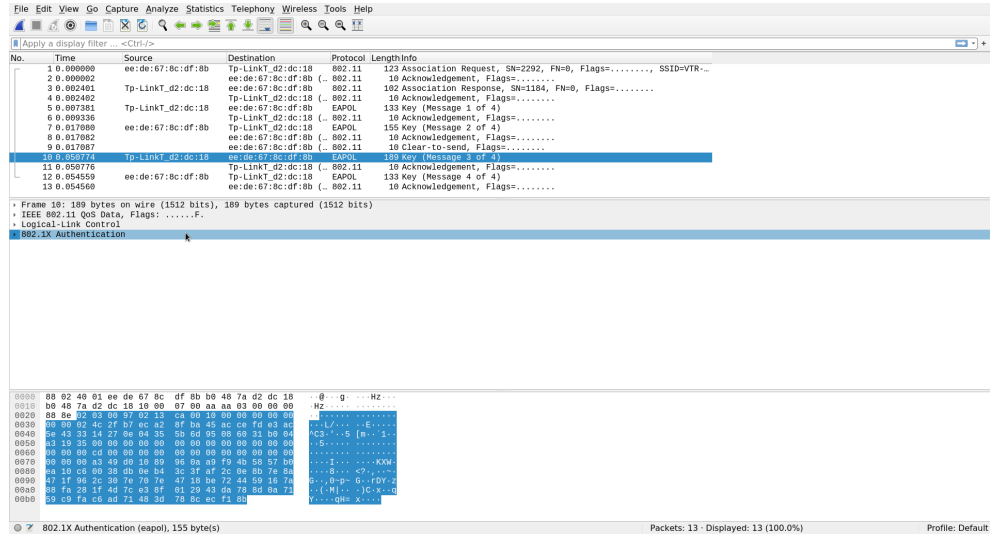


Figura 25: Data2 pedido por Pycrack.

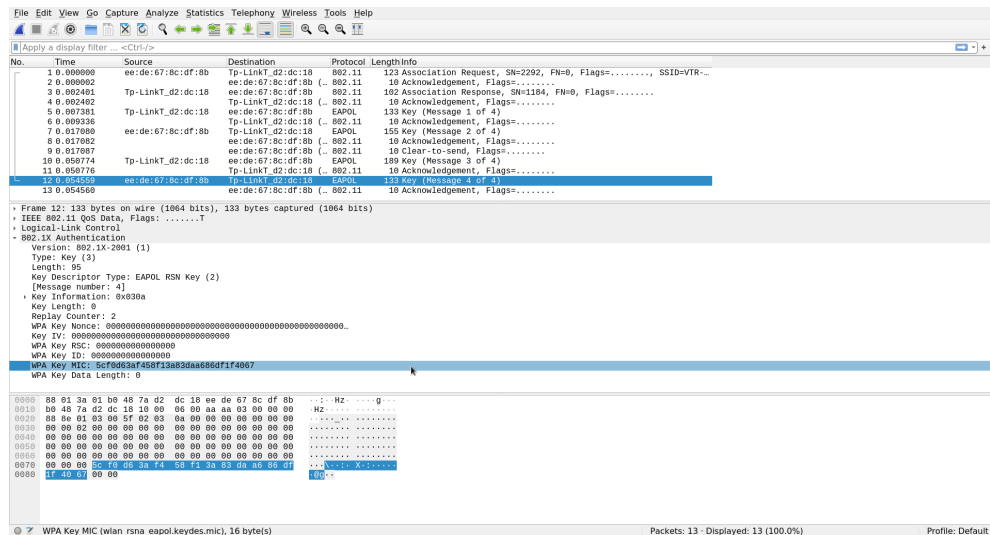


Figura 26: MIC3 pedido por Pycrack.

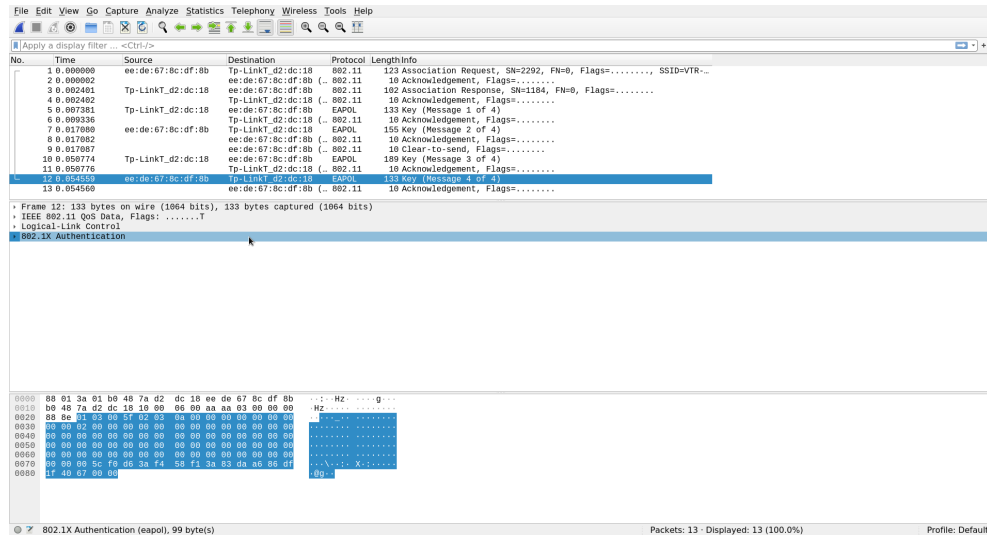


Figura 27: Data3 pedido por Pycrack.

## 4.7. Obtiene contraseña con pycrack

Una vez rellenados todos los parámetros pedidos en el script **pywd.py**, se procede a ejecutarlo utilizando **python3**:

```

~/PyCrack ➤ P master U:1 kerssen@kerZorinOS-B0HB-WAX9 18:35:51 python3 pywd.py
!!!Password Found!!!
Desired MIC1: 1813acb976741b446d43369fb96dbf90
Computed MIC1: 1813acb976741b446d43369fb96dbf90

Desired MIC2: a349d01089960aa9f94b5857b0ea10c6
Computed MIC2: a349d01089960aa9f94b5857b0ea10c6

Desired MIC2: 5cf0d63af458f13a83daa686df1f4067
Computed MIC2: 5cf0d63af458f13a83daa686df1f4067
Password: Security0

```

Figura 28: Uso de Pycrack.

Nuevamente, la contraseña encontrada es **Security0**.

## 5. Conclusiones y comentarios

Se puede concluir que se finalizó la experiencia de laboratorio de manera exitosa debido a que se lograron cumplir los objetivos planteados del mismo, identificado la red del informante, obteniendo la contraseña para descargar el archivo pedido y atacar a este último con un diccionario modificado utilizando 3 métodos diferentes.

Se exploraron varios conceptos fundamentales y herramientas utilizadas para el análisis y descifrado de tráfico en redes, reforzando el contenido visto en la cátedra y entendiendo el funcionamiento por detrás de cada una de las herramientas utilizadas.

### 5.1. Problemáticas en el Laboratorio

1. Al momento de buscar la página contenida en los paquetes enviados, no se podía encontrar debido a qué no se había utilizado el comando *airdecap* para poder descifrar los paquetes, por lo que se perdió tiempo en el laboratorio presencial analizando paquetes cifrados. La solución fue utilizar *airdecap* para descifrar los paquetes y así encontrar la página utilizando *Wireshark*.
2. Cuando se estaba intentando usar *hashcat* con potfile, no cargaba los hashes debido a que la extensión .pcap no era compatible para hacer el ataque. La solución fue utilizar la herramienta *hcxpcapngtool* para cambiar el formato a .hc22000, cargando sin problemas los hashes al potfile.
3. Nuevamente usando *hashcat*, el ataque no se realizaba debido al error *This OpenCL driver may fail kernel compilation or produce false negatives*. La solución implementada fue utilizar la flag *-force* para omitir el error y poder ejecutar *hashcat* con normalidad.
4. En la ejecución del script *pywd.py* de *Pycrack*, se tenía el error *UnicodeDecodeError: utf-8 codec cant decode byte 0xc0 in position 1020: invalid start byte*. La solución fue agregar *errors='ignore'* en donde se estaba leyendo el diccionario y se ejecutó con normalidad.