Informe Laboratorio 3

Sección 2

Vania Vergara
e-mail: vania.vergara@mail.udp.cl

Mayo de 2023

Índice

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 rockyou_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

La red es tenía mucha transmisión de datos y era la única red cifrada con WEP, además de tener el nombre del cifrado.

```
CH 12 ][ Elapsed: 6 mins ][ 2023-05-09 09:07 ][ PMKID found: 98:FC:11:86:B6:B9

BSSID              PWR  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

98:FC:11:86:B6:B9    0      507     9114    0   6  130   WPA2 CCMP   PSK  Telematica
48:D3:43:B9:F4:51   -1        0        0    0   6   -1                     <length:  0>
94:64:24:6B:85:23   -1        0        0    0  11   -1                     <length:  0>
00:72:78:1C:92:C1   -1        0        0    0   1   -1                     <length:  0>
B0:48:7A:D2:DD:74  -44     1080     2991    0   8   54e  WEP  WEP    SKA  WEP
7C:95:F3:C0:76:96  -54      118        0    0   1  195   OPN              WiFi_InvitadosUDP
```

Figura 1: Redes capturadas.

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

Airodump-ng captura paquetes cifrados enviados entre el punto de acceso y los dispositivos que se conectan a él, con el objetivo de descifrar la clave. Para lograr esto, se requiere capturar una cantidad suficiente de paquetes que puedan ser analizados por Aircrack-ng. Entonces, cuanto más compleja sea la clave, hará que se necesite más tiempo y paquetes.

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

Figura 2: Obtención de la llave.

2.4. indica el tiempo que demoró en obtener la password

No entregó el tiempo específico al obtener la llave, pero fueron milesimas de segundos.

2.5. descifra el contenido capturado

De los paquetes capturados del tráfico ICMP, los últimos bytes que están codificados en base64 corresponde a la url.

Figura 3: URL Wireshark.

Por lo tanto, se decodifica para obtener la url.

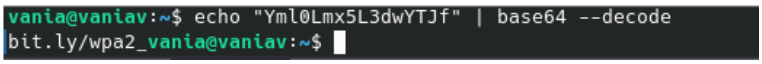

Figura 4: Obtención url.

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

De la figura 3 y 4 muestran los pasos para llegar a la url (https://www.cloudshark.org/captures/b5b39e1c51eb) donde se obtuvo el archivo handshake.pcap, que muestra tráfico de 4-way handshake.

3. Desarrollo (PASO 2)

3.1. indica script para modificar diccionario original

Para modificar el diccionario se implementó el siguiente código.

```python
with open("rockyou.txt", "r", encoding='latin-1') as archivo_origen, \
    open("rockyou_mod.dic", "w", encoding='latin-1') as archivo_destino:

    count = 0
    for linea in archivo_origen:

        # Si la línea comienza con un número, pasar a la siguiente línea
        if linea[0].isdigit():
            continue
        count += 1
        # Si la línea comienza con una letra, modificar la contraseña
        primera_letra = linea[0].upper()
        password = primera_letra + linea[1:].strip() + "0"

        archivo_destino.write(password + "\n")

print(f'El archivo modificado contiene {count} contraseñas.')

archivo_origen.close()
archivo_destino.close()
```

Figura 5: Script Python.

3.2. cantidad de passwords finales que contiene rockyou_mod.dic

```
vania@vanlav:~/Documentos/Universidad/Cripto/Lab3$ python3 rockyou_modificar.py
El archivo modificado contiene 11059798 contraseñas.
```

Figura 6: Número de contraseñas diccionario.

4. Desarrollo (Paso 3)

4.1. obtiene contraseña con hashcat con potfile

Para obtener la contraseña con hashcat, primeramente hay que pasar la captura .pcap a .hccapx. En este caso, se utilizó el siguiente link https://hashcat.net/cap2hashcat/.

```
vania@vaniav:~/Documentos/Universidad/Cripto/Lab3$ hashcat -m 22000 28385_1683649923.hc22000 rockyou_mo
d.dic
hashcat (v6.2.5) starting

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, 2859/5782 MB (1024 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

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: 2 MB

Dictionary cache hit:
* Filename..: rockyou_mod.dic
* Passwords.: 11059792
* Bytes.....: 119971529
* Keyspace..: 11059792

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

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 22000 (WPA-PBKDF2-PMKID+EAPOL)
Hash.Target......: 28385_1683649923.hc22000
Time.Started.....: Tue May  9 22:31:39 2023 (0 secs)
Time.Estimated...: Tue May  9 22:31:39 2023 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (rockyou_mod.dic)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:    12000 H/s (5.21ms) @ Accel:512 Loops:64 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests
Progress.........: 7203/11059792 (0.07%)
Rejected.........: 3107/7203 (43.13%)
Restore.Point....: 0/11059792 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: Password0 -> Trombone0
Hardware.Mon.#1..: Temp: 79c Util: 26%

Started: Tue May  9 22:31:36 2023
Stopped: Tue May  9 22:31:40 2023
```

Figura 7: Obtención contraseña con hashcat.

4.2. identifica nomenclatura del output

En el output se pueden ver distintos parámetros, como las características del dispositivo, la longitud mínima y máxima de la contraseña que es compatible con el kernel, la cantidad de memoria requerida para realizar el ataque, el diccionario utilizado, donde especifica la cantidad de contraseñas y el tamaño en bytes, información sobre el estado del ataque y las contraseñas recuperadas, que este caso corresponde a Security0.
Estas contraseñas se guardan en un potfile, donde si se vuelve a ejecutar lo mismo, indica que ya se encuentran descifradas y no necesitan ser procesadas de nuevo, como se muestra en la figura 8.

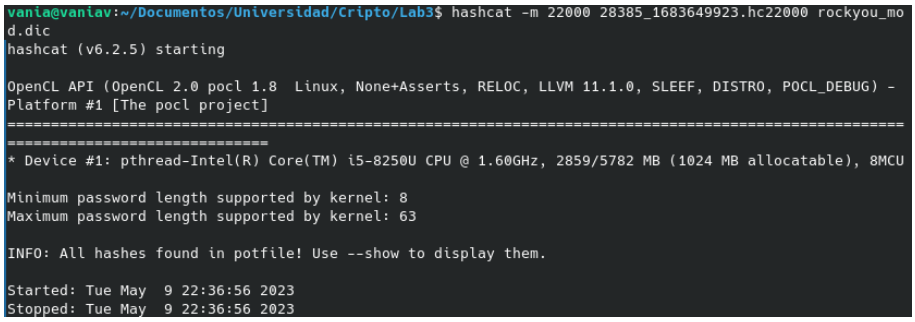

Figura 8: Obtención contraseña con hashcat.

4.3. obtiene contraseña con hashcat sin potfile

Para obtener la contraseña sin potfile, se agrega al final –potfile-disable.

```
vania@vaniav:~/Documentos/Universidad/Cripto/Lab3$ hashcat -m 22000 28385_1683649923.hc22000 rockyou_mo
d.dic --potfile-disable
hashcat (v6.2.5) starting

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, 2859/5782 MB (1024 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

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: 2 MB

Dictionary cache hit:
* Filename..: rockyou_mod.dic
* Passwords.: 11059792
```

```
Dictionary cache hit:
* Filename..: rockyou_mod.dic
* Passwords.: 11059792
* Bytes.....: 119971529
* Keyspace..: 11059792

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

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 22000 (WPA-PBKDF2-PMKID+EAPOL)
Hash.Target......: 28385_1683649923.hc22000
Time.Started.....: Tue May  9 21:59:48 2023 (1 sec)
Time.Estimated...: Tue May  9 21:59:49 2023 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (rockyou_mod.dic)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:    12137 H/s (4.98ms) @ Accel:512 Loops:64 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests
Progress.........: 7203/11059792 (0.07%)
Rejected.........: 3107/7203 (43.13%)
Restore.Point....: 0/11059792 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: Password0 -> Trombone0
Hardware.Mon.#1..: Temp: 77c Util: 19%

Started: Tue May  9 21:59:44 2023
Stopped: Tue May  9 21:59:50 2023
```

Figura 9: Obtención contraseña con hashcat.

4.4. identifica nomenclatura del output

En este caso, entrega la misma información entregada en el punto 4.2, ya que ignora el archivo de potfile y vuelve a procesar todo de nuevo.

4.5. obtiene contraseña con aircrack-ng

Para obtener la contraseña se ejecuto el siguiente comando: aircrack-ng -a2 -w rockyou_mod.dic handshake.pcap

Figura 10: Obtención contraseña con Aircrack-ng.

4.6. identifica y modifica parámetros solicitados por pycrack

Del la captura entregadas se puede ver el 4-way handshake, donde se obtienen los parámetros necesarios para el pycrack. Se necesita entregar el diccionario y modificar los campos de ssid, aNonce, sNonce, apMac, cliMac, data1, data2, data3, mic1, mic2 y mic3. En la figura 11 identifican los mensajes del handshake. Del primer mensaje, se obtiene aNonce (figura 12). Luego, del segundo mensaje, se puede identificar sNonce (figura 13). Las MAC del AP y del cliente se observan en la figura 14. Los campos mic1, mic2 y mic3 (WPA Key MIC) y data1, data2 y data3 se obtienen de los mensajes 2, 3 y 4 respectivamente. Los últimos corresponden al campo de 802.1X Authentication de wireshark (figura 15), donde cada WPA Key MIC está incluido dentro de data y debe reemplazarse por ceros. Por último, se puede identificar el ssid del primer mensaje, que se muestra en la figura 16.

No.	Time	Source	Destination	Protocol	Lengtl	Info
5	0.007381	Tp-LinkT_d2:dc:18	ee:de:67:8c:df:8b	EAPOL	133	Key (Message 1 of 4)
7	0.017080	ee:de:67:8c:df:8b	Tp-LinkT_d2:dc:18	EAPOL	155	Key (Message 2 of 4)
10	0.050774	Tp-LinkT_d2:dc:18	ee:de:67:8c:df:8b	EAPOL	189	Key (Message 3 of 4)
12	0.054559	ee:de:67:8c:df:8b	Tp-LinkT_d2:dc:18	EAPOL	133	Key (Message 4 of 4)

Figura 11: Handshake.

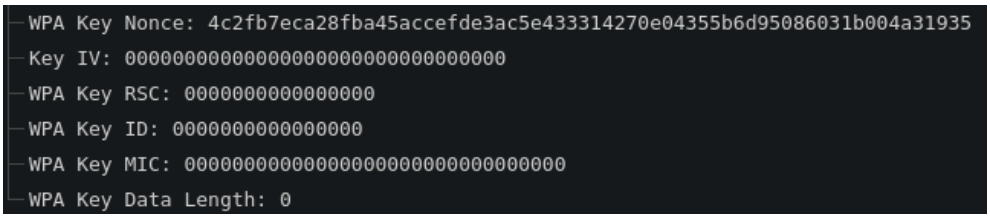

Figura 12: aNonce.

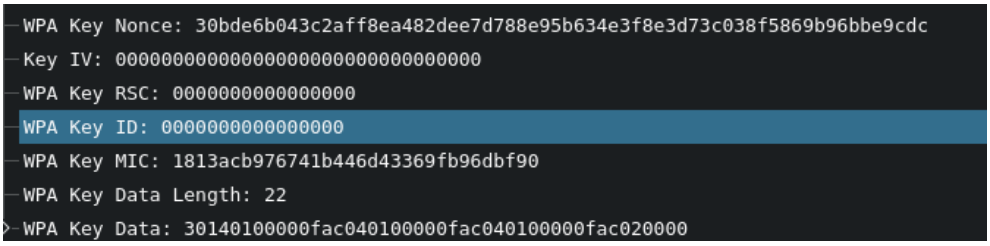

Figura 13: sNonce.

Figura 14: apMac y cliMac.

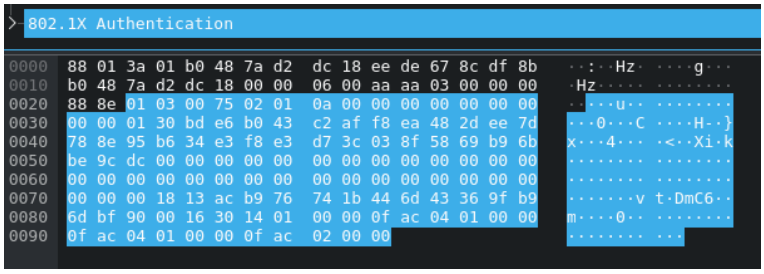

Figura 15: Sección de WPA2 EAPOL, paquete 2.

Figura 16: SSID.

Los cambios en el código se encuentran en la siguiente figura.

```
ssid = "VTR-1645213"
#ANonce
aNonce = a2b_hex('4c2fb7eca28fba45accefde3ac5e433314270e04355b6d95086031b004a31935')
#SNonce
sNonce = a2b_hex("30bde6b043c2aff8ea482dee7d788e95b634e3f8e3d73c038f5869b96bbe9cdc")
#Authenticator MAC (AP)
apMac = a2b_hex("b0487ad2dc18")
#Station address: MAC of client
cliMac = a2b_hex("eede678cdf8b")
#The first MIC
mic1 = "1813acb976741b446d43369fb96dbf90"
#The entire 802.1x frame of the second handshake message with the MIC field set to all zeros
data1 = a2b_hex
("0103007502010a000000000000000000130bde6b043c2aff8ea482dee7d788e95b634e3f8e3d73c038f5869b96bbe9cdc000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000001630140100000fac040100000fac040100000fac020000")
#The second MIC
mic2 = "a349d01089960aa9f94b5857b0ea10c6"
#The entire 802.1x frame of the third handshake message with the MIC field set to all zeros
data2 = a2b_hex
("02030970213ca001000000000000000024c2fb7eca28fba45accefde3ac5e433314270e04355b6d95086031b004a319350000000000000000000000000000
00000cd000000000000000000000000000000000000000000000000000000000000000000000038db0eb43c3faf2c0e8b7e8a471f962c307e707e4718be724459167a
88fa281f4d7ce38f012943da788d0a7159c9fac6ad71483d788cecf18b")
#The third MIC
mic3 = "5cf0d63af458f13a83daa686df1f4067"
#The entire 802.1x frame of the forth handshake message with the MIC field set to all zeros
data3 = a2b_hex
("0103005f02030a000000000000000000020000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000")
#Run an offline dictionary attack against the access point
TestPwds(S, ssid, aNonce, sNonce, apMac, cliMac, data1, data2, data3, mic1, mic2, mic3)
```

Figura 17: Parametros modificados pycrack.

4.7. obtiene contraseña con pycrack

```
vania@vaniav:~/Documentos/Universidad/Cripto/Lab3/PyCrack$ 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 18: Contraseña con pycrack.

Conclusiones y comentarios

Las herramientas vistas se pueden utilizar para realizar pruebas y descubrir posibles vulnerabilidades en los sistemas. La complejidad de las contraseñas influye en el tiempo que requiere emplear para obtenerlas.