

# Informe Laboratorio 3

## Sección 3

Alan Toro

e-mail: alan.toro@mail.udp.cl

Octubre de 2023

## Índice

<b>1. Descripción de actividades</b>	<b>2</b>
<b>2. Desarrollo (PASO 1)</b>	<b>2</b>
2.1. Identificar 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 . . . . .	3
2.3. Obtiene la password con ataque por defecto de aircrack-ng . . . . .	4
2.4. Indica el tiempo que demoró en obtener la password . . . . .	5
2.5. Descifra el contenido capturado . . . . .	5
2.6. Describe como obtiene la url de donde descargar el archivo . . . . .	5
<b>3. Desarrollo (PASO 2)</b>	<b>6</b>
3.1. Indica script para modificar diccionario original . . . . .	6
3.2. Cantidad de passwords finales que contiene rockyou_mod.dic . . . . .	6
<b>4. Desarrollo (Paso 3)</b>	<b>7</b>
4.1. Obtiene contraseña con hashcat con potfile . . . . .	7
4.2. Identifica nomenclatura del output . . . . .	9
4.3. Obtiene contraseña con hashcat sin potfile . . . . .	9
4.4. Identifica nomenclatura del output . . . . .	11
4.5. Obtiene contraseña con aircrack-ng . . . . .	11
4.6. Identifica y modifica parámetros solicitados por pycrack . . . . .	12
4.7. Obtiene contraseña con pycrack . . . . .	13

## 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)

La siguiente actividad será realizada usando un computador con Garuda Linux (Linux-Zen 6.5.8) con una tarjeta de red integrada (Intel Cannon Point-LP CNVi). Además, para la ejecución de esta se usa la tarjeta de red en modo monitor. Para esto se utiliza el comando `sudo airmon check` para comprobar los servicios que están haciendo uso de la tarjeta de red.

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

```
wzrdd@heth in repo: cripto/Lab3/informe on P main [!?] as 🐼 took 142ms
λ sudo airmon-ng check

Found 4 processes that could cause trouble.
Kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode

PID Name
633 avahi-daemon
654 NetworkManager
655 wpa_supplicant
657 avahi-daemon
```

Figura 1: Procesos que podrían generar conflictos para iniciar el modo monitor.

Se usan 3 comandos para detener estos procesos.

```
1 sudo systemctl stop wpa_supplicant
2 sudo systemctl stop NetworkManager
3 sudo systemctl disable --now avahi-daemon
```

Luego de esto, identificamos la tarjeta de red wifi a utilizar para todo el laboratorio utilizando `iwconfig`, a lo que identificamos la interfaz **wlp0s20f3** y bastaría con usar el comando `sudo airmon-ng start wlp0s20f3` para colocar la tarjeta en modo monitor, esto cambia el nombre de la interfaz a **wlp0s20f3mon**.

```
wzrdd@heth in repo: cripto/Lab3/informe on P main [!?] as 🐼 took 353ms
λ sudo airmon-ng start wlp0s20f3

PHY      Interface      Driver      Chipset
phy0     wlp0s20f3      iwlwifi     Intel Corporation Cannon Point-LP CNVi [Wireless-AC] (rev 11)
          (mac80211 monitor mode vif enabled for [phy0]wlp0s20f3 on [phy0]wlp0s20f3mon)
          (mac80211 station mode vif disabled for [phy0]wlp0s20f3)
```

Figura 2: Correcto cambio a modo monitor de la interfaz wlp0s20f3.

### 2.1. Identificar en qué se destaca la red del informante del resto

Para listar las redes disponibles, se usa el comando `nmcli dev wifi`, dentro de la lista de redes se nota una única red con cifrado WEP, protocolo obsoleto, inseguro y atacable.

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

El ataque para obtener la pass es un ataque de colisión por fuerza bruta, en particular un ataque de cumpleaños (*Birthday Attack*), en este se dice que la probabilidad de encon-

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

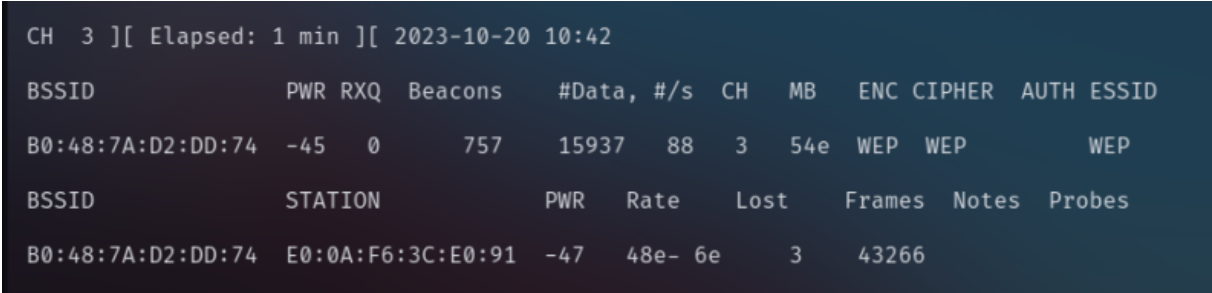
trar paquete con un vector de inicialización duplicado desde  $\sqrt{n}$  donde  $n$  es la cantidad de vectores posibles comienza a ser del 50 %. Se sabe que el protocolo WEP utiliza vectores de inicialización de 24 bits, por lo que  $n = 2^{24}$ , lo que implica que desde  $\sqrt{2^{24}} = 4,096$  paquetes la probabilidad de encontrar un paquete duplicado comienza a ser del 50 %. Es por esto que un buen número para comenzar a romper la contraseña es del orden de los 5.000 vectores de inicialización.

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

Para obtener los IV, identificamos que la red WEP se encuentra en el canal 3 y su bssid corresponde a B0:48:7A:D2:DD:74. Con esto podemos dirigir el ataque directamente hacia la red víctima. Esto se realiza con el comando:

```
1 sudo airodump-ng -c 3 --bssid B0:48:7A:D2:DD:74 -w dump wlp0s20f3mon
```

En donde la flag `-c 3` indica el canal y `--bssid B0:48:7A:D2:DD:74` indican el canal de la víctima y la dirección física. El flag `-w dump` genera un output en archivos `dump.csv` y `dump.cap` que serán utilizados por `aircrack-ng` para obtener la pass. Finalmente se indica la interfaz de red `wlp0s20f3mon`.



CH 3 ][ Elapsed: 1 min ][ 2023-10-20 10:42										
BSSID	PWR	RXQ	Beacons	#Data	#/s	CH	MB	ENC	CIPHER	AUTH ESSID
B0:48:7A:D2:DD:74	-45	0	757	15937	88	3	54e	WEP	WEP	WEP
BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes			
B0:48:7A:D2:DD:74	E0:0A:F6:3C:E0:91	-47	48e- 6e	3	43266					

Figura 3: Comando airodump dirigido hacia la víctima capturando IVs.

Se logra obtener la contraseña con más de 20.000 IVs. En particular con 23.034 IVs. El comando airodump anteriormente descrito genera un archivo `dump-01.cap`, este archivo se usa como entrada para ejecutar el comando:

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

Este comando cada 5.000 IVs intenta crackear la pass. Finalmente en los 23.034 se logra esto:

```

AirCrack-ng 1.7

[00:00:01] Tested 26500 keys (got 23034 IVs)

KB    depth  byte(vote)
0 0/ 12 12(30720) 93(28928) 7B(28416) 81(28416) 4D(27904) 52(27648) F1(27648) 59(27392) DD(27392) 79(27136) 85(27136) AC(26880) 2A(26624) 2B(26624) 32(26624)
1 0/ 1 34(34816) 42(29696) 53(28416) 1E(28160) D4(28160) F7(28160) 1F(27904) 32(27904) 4F(27904) 57(27904) 61(27904) 9B(27904) BB(27648) E4(27648) 8B(27392)
2 0/ 23 56(30464) 94(29184) A6(28928) 02(28416) 3E(28160) 1B(27904) ED(27648) 57(27392) 87(27392) 90(27392) 99(27392) F2(27136) 19(27136) 4C(27136) 6B(27136)
3 2/ 32 78(28672) AF(28672) 21(28160) B5(28160) 7C(27904) CF(27648) F5(27648) 11(27392) 37(27392) 1D(27136) 49(27136) 68(27136) 0B(26880) 90(26880) EA(26880)
4 2/ 4 98(29952) A4(28928) D7(28672) 62(28416) BF(28416) 08(27904) 82(27904) 13(27648) 1C(27648) 74(27392) 09(27136) 64(26880) 68(26880) 1B(26624) 24(26624)

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

```

Figura 4: Comando aircrack-ng con la contraseña encontrada.

La contraseña de la red es **12:34:56:78:90**.

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

El tiempo de captura total fue de 2 minutos y 7 segundos. Esto se puede verificar en el archivo dump-01.csv entonces la hora de inicio de captura fue en 2023-10-20 10:40:50 y el tiempo de término fue 2023-10-20 10:42:57. Además, con los IVs ya capturados crackear la contraseña tomó 1 segundo, esto se puede ver en la figura 4 de la subsección anterior.

## 2.5. Descifra el contenido capturado

Para decifrar el tráfico se utiliza el comando `sudo airedecap-ng -w 12:34:56:78:90 dump-01.cap` en donde la flag `-w 12:34:56:78:90` corresponde a la contraseña encontrada y dump-01.cap al archivo de la captura de airodump de la sección anterior. El resultado de este comando es la generación del archivo **dump-01-dec.cap**.

```

AirCrack-ng 1.7

[00:00:01] Tested 26500 keys (got 23034 IVs)

KB    depth  byte(vote)
0 0/ 12 12(30720) 93(28928) 7B(28416) 81(28416) 4D(27904) 52(27648) F1(27648) 59(27392) DD(27392) 79(27136) 85(27136) AC(26880) 2A(26624) 2B(26624) 32(26624)
1 0/ 1 34(34816) 42(29696) 53(28416) 1E(28160) D4(28160) F7(28160) 1F(27904) 32(27904) 4F(27904) 57(27904) 61(27904) 9B(27904) BB(27648) E4(27648) 8B(27392)
2 0/ 23 56(30464) 94(29184) A6(28928) 02(28416) 3E(28160) 1B(27904) ED(27648) 57(27392) 87(27392) 90(27392) 99(27392) F2(27136) 19(27136) 4C(27136) 6B(27136)
3 2/ 32 78(28672) AF(28672) 21(28160) B5(28160) 7C(27904) CF(27648) F5(27648) 11(27392) 37(27392) 1D(27136) 49(27136) 68(27136) 0B(26880) 90(26880) EA(26880)
4 2/ 4 98(29952) A4(28928) D7(28672) 62(28416) BF(28416) 08(27904) 82(27904) 13(27648) 1C(27648) 74(27392) 09(27136) 64(26880) 68(26880) 1B(26624) 24(26624)

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

```

Figura 5: Comando airedecap decifrando los paquetes.

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

Finalmente, se analizan los paquetes capturados usando Wireshark. A inspección simple se encuentra un flujo constante de paquetes ICMP, en donde, todos en el payload tienen una URL.

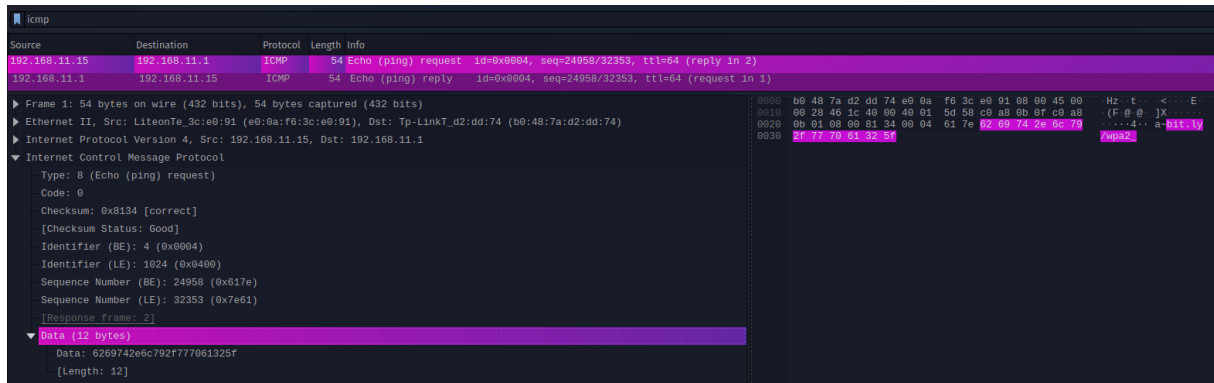


Figura 6: Payload de un paquete ICMP capturado en donde se encuentra la URL.

La URL redirige hacia <https://www.cloudshark.org/captures/b5b39e1c51eb> en donde se encuentra la captura de un handshake.

### 3. Desarrollo (PASO 2)

#### 3.1. Indica script para modificar diccionario original

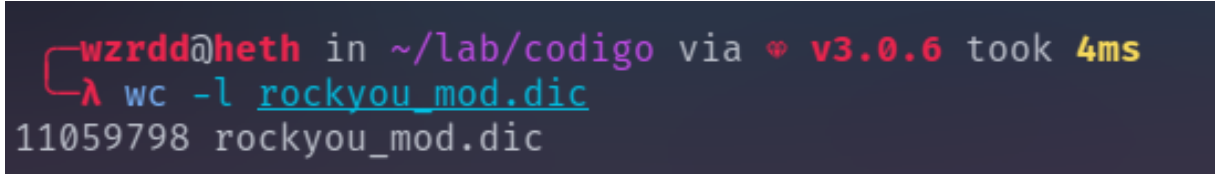
Para generar un diccionario modificado se usa un script escrito en Ruby. Adjunto en el proyecto como `codigos/modify_dict.rb` que también se muestra a continuación:

```
1 File.open("rockyou.txt", :encoding => 'iso-8859-15') do |input_file|
2   File.open("rockyou_mod.dic", "w") do |output_file|
3     input_file.each_line do |line|
4       if !line.match?(/\A\d/)
5         output_file.puts line.capitalize.strip + '0'
6       end
7     end
8   end
9 end
```

Lo que genera un nuevo diccionario llamado `rockyou_mod.dic`.

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

Cada contraseña es una columna por lo que se cuentan la cantidad de filas del archivo generado usando `wc -l`. Lo indica 11.059.798 de contraseñas finales.



```
wzrdd@heth in ~/lab/codigo via v3.0.6 took 4ms  
λ wc -l rockyou_mod.dic  
11059798 rockyou_mod.dic
```

Figura 7: Cantidad de passwords finales.

## 4. Desarrollo (Paso 3)

Con el archivo encontrado en el paso uso **handshake.pcap** convertimos a formato hc22000 para recuperar una PSK. Para esto usamos el comando:

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

### 4.1. Obtiene contraseña con hashcat con potfile

Para obtener la contraseña usando hashcat se usa el comando:

```
1 hashcat --potfile-path potfile.txt -m 22000 handshake.hc22000  
rockyou_mod.dic
```

Lo que genera el output:

```
hashcat --potfile-path potfile.txt -m 22000 handshake.hc22000 rockyou_mod.dic
hashcat (v6.2.6) starting

OpenCL API (OpenCL 2.1 LINUX) - Platform #1 [Intel(R) Corporation]

* Device #1: Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz, 7805/15675 MB (1959 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 built:
* Filename..: rockyou_mod.dic
* Passwords.: 11059798
* Bytes.....: 119975893
* Keyspace..: 11059791
* Runtime...: 1 sec

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

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 22000 (WPA-PBKDF2-PMKID+EAPOL)
Hash.Target.....: handshake.hc22000
Time.Started.....: Sun Oct 22 07:05:57 2023 (0 secs)
Time.Estimated...: Sun Oct 22 07:05:57 2023 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (rockyou_mod.dic)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 10892 H/s (11.41ms) @ Accel:256 Loops:256 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 3817/11059791 (0.03%)
Rejected.....: 1769/3817 (46.35%)
Restore.Point....: 0/11059791 (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: 96c Util: 82%

Started: Sun Oct 22 07:05:40 2023
Stopped: Sun Oct 22 07:05:59 2023
```

Figura 8: Output hashcat con potfile.

La contraseña es SECURITY0 y el SSID es VTR-1645213, ambos se encuentran también en el potfile.



## 4.2. Identifica nomenclatura del output

- Device: Describe la unidad de procesamiento encargada de ejecutar la fuerza bruta. En este caso fue una CPU integrada Intel i5-8265U.
- Se describe el mínimo y máximo largo que se puede procesar por el kernel. En este caso 8 mínimo y 63 máximo.
- Se detalla la cantidad de hashes y describen los bitmaps.
- Se indica la lista de optimizadores utilizados.
- Watchdog determina una temperatura máxima del device, si se supera se aborta la tarea.
- Se describe el cache del diccionario.
- Se listan los hash y credenciales *crackeados*.
- Entre los más importantes que siguen está: el estado como crackeado, el modo de hash, los tiempos de inicio y término, la velocidad de hasheado y los candidatos siguientes.

## 4.3. Obtiene contraseña con hashcat sin potfile

Para obtener la contraseña usando hashcat sin potfile se usa la flag `--potfile-disable`. El comando a usar queda como:

```
1 hashcat --potfile-disable -m 22000 handshake.hc22000 rockyou_mod.dic
```

Lo que genera el output:

```
hashcat --potfile-disable -m 22000 handshake.hc22000 rockyou_mod.dic
hashcat (v6.2.6) starting

OpenCL API (OpenCL 2.1 LINUX) - Platform #1 [Intel(R) Corporation]

* Device #1: Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz, 7805/15675 MB (1959 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.: 11059791
* Bytes.....: 119975893
* Keyspace..: 11059791

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

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 22000 (WPA-PBKDF2-PMKID+EAPOL)
Hash.Target.....: handshake.hc22000
Time.Started.....: Sun Oct 22 07:38:16 2023 (0 secs)
Time.Estimated...: Sun Oct 22 07:38:16 2023 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (rockyou_mod.dic)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 12418 H/s (9.91ms) @ Accel:64 Loops:1024 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 2907/11059791 (0.03%)
Rejected.....: 1371/2907 (47.16%)
Restore.Point....: 1965/11059791 (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: 88c Util: 18%

Started: Sun Oct 22 07:38:13 2023
Stopped: Sun Oct 22 07:38:18 2023
```

Figura 9: Output hashcat sin potfile.

Se llega a la misma contraseña y SSID que la sección anterior con potfile.

#### 4.4. Identifica nomenclatura del output

- Device: Describe la unidad de procesamiento encargada de ejecutar la fuerza bruta. En este caso fue una CPU integrada Intel i5-8265U.
- Se describe el mínimo y máximo largo que se puede procesar por el kernel. En este caso 8 mínimo y 63 máximo.
- Se detalla la cantidad de hashes y describen los bitmaps.
- Se indica la lista de optimizadores utilizados.
- Watchdog determina una temperatura máxima del device, si se supera se aborta la tarea.
- Se describe el cache del diccionario.
- Se listan los hash y credenciales *crackeados*.
- Entre los más importantes que siguen está: el estado como crackeado, el modo de hash, los tiempos de inicio y término, la velocidad de hasheado y los candidatos siguientes.

#### 4.5. Obtiene contraseña con aircrack-ng

Para encontrar la contraseña usando aircrack-ng se usa el comando:

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

Lo que genera el output

```

Aircrack-ng 1.7

[00:00:00] 3439/9296278 keys tested (11269.42 k/s)

Time left: 13 minutes, 44 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 10: Output aircrack encontrando la contraseña.

También converge a la contraseña SECURITY0.

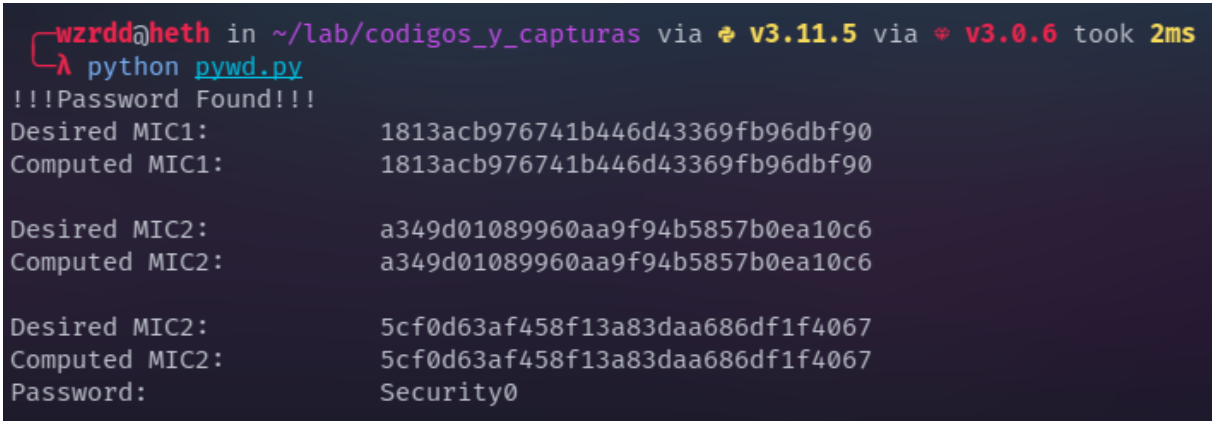
## 4.6. Identifica y modifica parámetros solicitados por pycrack

Los parámetros a modificar son:

- Path del archivo al diccionario.
- El SSID que se puede encontrar en el primer paquete o desde el output de hashcat de la sección anterior.
- El Authenticator y Supplicant Nounce. Estos se pueden encontrar en el primer y segundo mensaje del protocolo EAPOL respectivamente. En el apartado “802.1X Authentication”, en el campo “WPA Key Nonce” de cada uno.
- Las direcciones físicas del Authenticator y Supplicant, se puede encontrar en todos los paquetes EAPOL.
- El MIC1 se encuentra en el segundo paquete del protocolo EAPOL, el MIC2 en el tercer paquete EAPOL y el MIC se encuentra en el cuarto paquete.
- Los campos DATA1, 2 y 3 son el bloque completo de “802.1X Authentication” como hexadecimal.

Todos los cambios están comentados al final de la línea como un “CAMBIADO” para identificarlos en el código pywd.py adjunto en el proyecto.

## 4.7. Obtiene contraseña con pycrack

A terminal window with a dark background. The prompt is 'wzrdd@heth' in red. The command 'python pywd.py' is entered in blue. The output shows '!!!Password Found!!!' in green. Below this, there are three pairs of 'Desired MIC' and 'Computed MIC' values, all matching. The final line shows 'Password: Security0' in green.

```
wzrdd@heth in ~/lab/codigos_y_capturas via v3.11.5 via v3.0.6 took 2ms
λ python 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 11: Output pycrack encontrando la contraseña

Al igual que los otros métodos, encuentra la contraseña SECURITY0.

## Conclusiones y comentarios

Primero que todo, la conclusión más importante es el poco tiempo que tomó quebrar todas las seguridades. En el caso de WEP se logró encontrar una contraseña segura de largo 15 en un tiempo aproximado de 2 minutos. Esto quiere decir que contraseñas seguras sobre protocolos inseguros no mejoran la seguridad ya que la seguridad se pierde por el hilo más fino. En este caso particular un ataque de colisiones por fuerza bruta es suficiente para exponer la red.

Con respecto a los siguientes pasos los tiempos también fueron extremadamente cortos en un computador de recursos medios. Es por esto que para auditar la seguridad de una red valga la pena revisar diccionarios comunes y variantes que se pueden lograr a partir de estos. Ya que, un ataque por diccionario a un tráfico interceptado puede exponer la red de manera rápida y fácil incluso utilizando protocolos seguros como WPA.

## Enlace a Github

- <https://github.com/wzrdd/cripto>