

Informe Laboratorio 3

Sección 1

Sebastián Riquelme
e-mail: sebastian.riquelme1@mail.udp.cl

Mayo de 2023

Índice

1. Descripción de actividades	2
2. Desarrollo (PASO 1)	2
2.1. identificar en qué se destaca la red del informante del resto	2
2.2. explica matemáticamente porqué se requieren más de 5000 paquetes para obtener la pass	2
2.3. obtiene la password con ataque por defecto de aircrack-ng	3
2.4. indica el tiempo que demoró en obtener la password	3
2.5. descifra el contenido capturado	4
2.6. describe como obtiene la url de donde descargar el archivo	4
3. Desarrollo (PASO 2)	4
3.1. indica script para modificar diccionario original	4
3.2. cantidad de passwords finales que contiene rockyou_mod.dic	5
4. Desarrollo (Paso 3)	5
4.1. obtiene contraseña con hashcat con potfile	5
4.2. Identificación de la nomenclatura del output	5
4.3. Obtiene contraseña con hashcat sin potfile	7
4.4. Identifica nomenclatura del output	7
4.5. obtiene contraseña con aircrack-ng	8
4.6. identifica y modifica parámetros solicitados por pycrack	8
4.7. Descripción de los parámetros del script	9
4.8. obtiene contraseña con pycrack	10

1. Descripción de actividades

Su informante quiere entregarle la contraseña de acceso a una red, pero desconfía de todo medio para entregársela (aún no llega al capítulo del curso en donde aprende a comunicar una password sin que nadie más la pueda interceptar). Por lo tanto, le entregará un archivo que contiene un desafío de autenticación, que al analizarlo, usted podrá obtener la contraseña que lo permite resolver. Como nadie puede ver a su informante (es informante y debe mantener el anonimato), él se comunicará con usted a través de la redes inalámbricas y de una forma que solo usted, como experto en informática y telecomunicaciones, logrará esclarecer.

1. Identifique cual es la red inalámbrica que está utilizando su informante para enviarle información. Obtenga la contraseña de esa red utilizando el ataque por defecto de aircrack-ng, indicando el tiempo requerido para esto. Descifre el contenido transmitido sobre ella y descargue de Internet el archivo que su informante le ha comunicado a través de los paquetes que usted ha descifrado.
2. Descargue el diccionario de RockyouLinks to an external site. (utilizado ampliamente en el mundo del pentesting). Haga un script que para cada string contenido en el diccionario, reemplace la primera letra por su letra en capital y agregue un cero al final de la password.
3. Todos los strings que comiencen con número toca eliminarlos del diccionario. Indique la cantidad de contraseñas que contiene el diccionario modificado debe llamarse rock-you_mod.dic A continuación un ejemplo de cómo se modifican las 10 primeras líneas del diccionario original.

2. Desarrollo (PASO 1)

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

La red utilizada por el informante exhibía una característica notablemente distintiva: estaba cifrada mediante el protocolo WEP. Esta forma de seguridad es prácticamente obsoleta y raramente utilizada en la actualidad, lo cual la hacía resaltar entre las demás. Esta peculiaridad facilitó considerablemente su identificación.

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

La razón principal de por qué necesitamos más de 5000 IVs (Valores de Inicialización) para realizar un ataque exitoso se debe a la naturaleza de la criptografía de la clave WEP y cómo aircrack-ng ataca este sistema.

La clave WEP está formada por una combinación de una clave secreta y un valor de inicialización de 24 bits. Este último se transmite en texto plano, lo que significa que puede ser interceptado. Sin embargo, la clave secreta permanece oculta y es la que queremos descubrir.

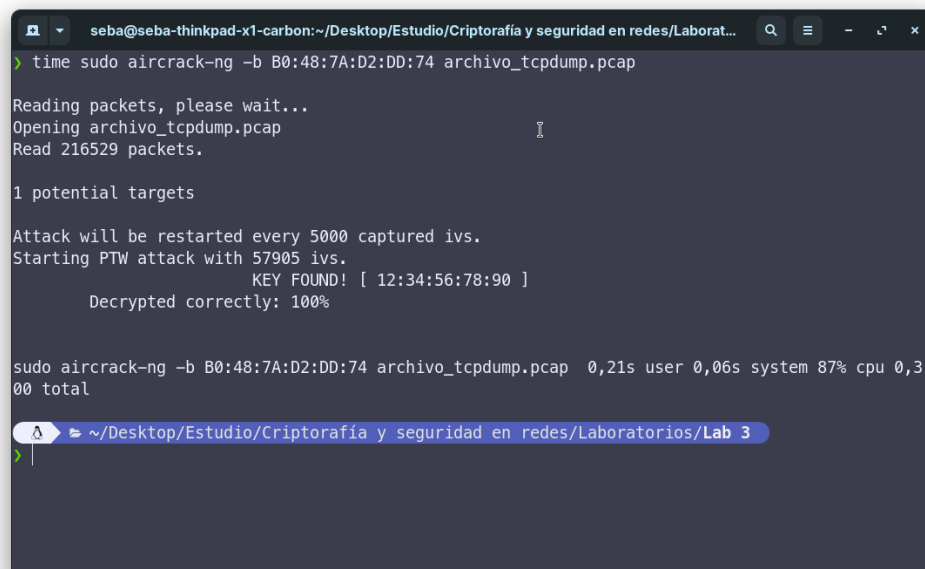
2.3 obtiene la password con ataque por defecto de aircrack-ng DESARROLLO (PASO 1)

Aircrack-ng utiliza un ataque estadístico conocido como Ataque de FMS (por sus creadores Fluhrer, Mantin y Shamir) para descifrar la clave WEP. Este ataque se basa en debilidades en la generación de la secuencia de cifrado RC4 utilizada por WEP, y específicamente en cómo esta secuencia de cifrado está correlacionada con el Valor de Inicialización.

El Ataque de FMS necesita una gran cantidad de IVs para tener suficientes datos para analizar y encontrar la correlación que revela la clave secreta. Matemáticamente, se puede demostrar que la cantidad de IVs necesarios para este ataque es aproximadamente de 5000 a 10000, aunque en la práctica, normalmente se necesitan muchos más debido a los paquetes corruptos, el ruido en la señal y otros factores.

Entonces, la necesidad de más de 5000 IVs no es un requisito estricto, pero es una estimación de cuántos se necesitan en promedio para realizar un ataque exitoso usando aircrack-ng y el Ataque de FMS.

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



```
seba@seba-thinkpad-x1-carbon:~/Desktop/Estudio/Criptografía y seguridad en redes/Laborat...
> time sudo aircrack-ng -b B0:48:7A:D2:DD:74 archivo_tcpdump.pcap

Reading packets, please wait...
Opening archivo_tcpdump.pcap
Read 216529 packets.

1 potential targets

Attack will be restarted every 5000 captured ivs.
Starting PTW attack with 57905 ivs.
KEY FOUND! [ 12:34:56:78:90 ]
Decrypted correctly: 100%

sudo aircrack-ng -b B0:48:7A:D2:DD:74 archivo_tcpdump.pcap 0,21s user 0,06s system 87% cpu 0,300 total

~/Desktop/Estudio/Criptografía y seguridad en redes/Laboratorios/Lab 3
>
```

Figura 1: Password obtenida con ataque por defecto de aircrack-ng.

2.4. indica el tiempo que demoró en obtener la password

Tardó 0,21 segundos como se aprecia en la Figura 1.

2.5. descifra el contenido capturado

Una vez obtenida la contraseña WEP de la red inalámbrica, es posible descifrar el contenido de la captura de paquetes realizada en dicha red. Para esto, se utilizó la herramienta Wireshark, que permite el análisis de paquetes de red y, en este caso, el descifrado de los mismos al contar con la clave de seguridad WEP.

Los pasos seguidos fueron los siguientes:

En primer lugar, se abrió Wireshark y se cargó el archivo de captura de paquetes obtenido previamente.

Seguidamente, se accedió al menú Edit y se seleccionó la opción Preferences.

Dentro de las preferencias, se navegó hasta Protocols ¿IEEE 802.11. Aquí se encuentran las opciones de configuración para el protocolo de red inalámbrica.

En la casilla Decryption Keys, se hizo click en el botón Edit.

Se pulsó en el botón New y se seleccionó WEP en el desplegable de tipo de clave. En la casilla correspondiente, se introdujo la clave WEP obtenida previamente.

Con la clave ya introducida, se confirmaron los cambios y se cerraron las ventanas de configuración.

Al volver a la ventana principal de Wireshark, la captura de paquetes se mostró ahora de manera descifrada. En este punto, se pudo proceder al análisis detallado del tráfico de red capturado.

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

La URL desde la que se debía descargar el archivo fue extraída a partir del contenido capturado de los paquetes de red. Este contenido estaba codificado en Base64 y se encontraba dentro del payload de los paquetes ICMP.

La decodificación de Base64 es un procedimiento estándar que se puede llevar a cabo utilizando diversas herramientas o lenguajes de programación. En este caso, tras decodificar el contenido del payload, se reveló la URL desde la que se podía descargar el archivo requerido.

La URL obtenida tras el proceso de decodificación fue la siguiente: `http://bit.ly/wpa2_`. Con esta información, se pudo proceder a la descarga del archivo correspondiente para continuar con las tareas del laboratorio.

3. Desarrollo (PASO 2)

3.1. indica script para modificar diccionario original

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



```
1 # Importamos el módulo necesario
2 import re
3
4 # Nombre de los archivos de entrada y salida
5 input_file = "rockyou.txt"
6 output_file = "rockyou_mod.dic"
7
8 # Contador para las contraseñas modificadas
9 count = 0
10
11 # Abrimos los archivos de entrada y salida
12 with open(input_file, "r", encoding="latin-1") as infile, open(output_file, "w", encoding="latin-1") as outfile:
13     for line in infile:
14         # Eliminamos los espacios en blanco al principio y al final
15         line = line.strip()
16         # Verificamos si la línea comienza con una letra
17         if re.match(r'^[a-zA-Z]', line):
18             # Modificamos la primera letra y agregamos un cero al final
19             line = line[0].upper() + line[1:] + '0'
20             # Escribimos la línea en el archivo de salida
21             outfile.write(line + '\n')
22             # Incrementamos el contador
23             count += 1
24
25 # Imprimimos el número de contraseñas en el diccionario modificado
26 print(f"El diccionario modificado contiene {count} contraseñas.")
```

Figura 2: Script para modificar rockyou.

3.2. cantidad de passwords finales que contiene rockyou_mod.dic

El diccionario modificado contiene 10.957.144 contraseñas.

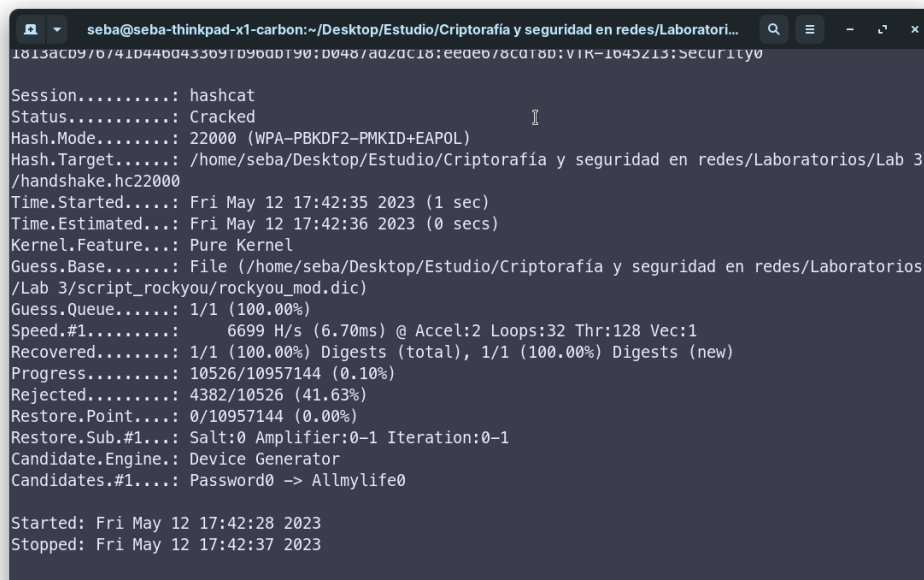
4. Desarrollo (Paso 3)

4.1. obtiene contraseña con hashcat con potfile

Crack con potfile.

```
"/home/seba/Desktop/Estudio/Criptoraf a y seguridad en redes/
Laboratorios/Lab 3/hashcat/hashcat-6.2.6/hashcat.bin" -m 22000 -
a 0 "/home/seba/Desktop/Estudio/Criptoraf a y seguridad en
redes/Laboratorios/Lab 3/handshake.hc22000" "/home/seba/Desktop/
Estudio/Criptoraf a y seguridad en redes/Laboratorios/Lab 3/
script_rockyou/rockyou_mod.dic"
```

4.2. Identificación de la nomenclatura del output



```
1813ACD9/0/41D4400433091D90DD190:0048/a020c18:eedeb/8cd180:VIR-1045213:Security0

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 22000 (WPA-PBKDF2-PMKID+EAPOL)
Hash.Target.....: /home/seba/Desktop/Estudio/Criptografía y seguridad en redes/Laboratorios/Lab 3
/handshake.hc22000
Time.Started....: Fri May 12 17:42:35 2023 (1 sec)
Time.Estimated...: Fri May 12 17:42:36 2023 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (/home/seba/Desktop/Estudio/Criptografía y seguridad en redes/Laboratorios
/Lab 3/script_rockyou/rockyou_mod.dic)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 6699 H/s (6.70ms) @ Accel:2 Loops:32 Thr:128 Vec:1
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 10526/10957144 (0.10%)
Rejected.....: 4382/10526 (41.63%)
Restore.Point....: 0/10957144 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: Password0 -> Allmylife0

Started: Fri May 12 17:42:28 2023
Stopped: Fri May 12 17:42:37 2023
```

Figura 3: Output hashcat.

La Figura 3 muestra la salida de información de estado de hashcat. Los elementos de esta salida son los siguientes:

- **Session:** Nombre de la sesión de hashcat.
- **Status:** Estado de la ejecución de hashcat.
- **Hash.Mode:** El modo de hash usado, correspondiente a la opción -m en la línea de comandos.
- **Hash.Target:** El hash o archivo de hash que se está atacando.
- **Time.Started:** Momento en que comenzó la ejecución de hashcat.
- **Time.Estimated:** Estimación de cuándo finalizará hashcat.
- **Guess.Base:** Base de la suposición de la contraseña.
- **Guess.Mod:** Modificador de suposiciones, si se utiliza.
- **Speed.#1, Speed.#2, Speed.##*:** Tasas de hash en diferentes dispositivos y la tasa de hash total.
- **Recovered:** Cantidad de hashes que se han descifrado.
- **Progress:** Cantidad de suposiciones que se han hecho.

- **Rejected:** Cantidad de suposiciones que han sido rechazadas.
- **Restore.Point:** Cantidad de puntos de restauración procesados.
- **Candidates.#1, Candidates.#2, Candidates.#*:** Contraseñas candidatas que se están probando actualmente en diferentes dispositivos.

4.3. Obtiene contraseña con hashcat sin potfile

En este apartado se ilustra cómo obtener la contraseña con hashcat sin utilizar el archivo potfile. El comando utilizado es el siguiente:

```
"/home/seba/Desktop/Estudio/Criptoraf a y seguridad en redes/  
Laboratorios/Lab 3/hashcat/hashcat-6.2.6/hashcat.bin" -m 22000 -  
a 0 "/home/seba/Desktop/Estudio/Criptoraf a y seguridad en  
redes/Laboratorios/Lab 3/handshake.hc22000" "/home/seba/Desktop/  
Estudio/Criptoraf a y seguridad en redes/Laboratorios/Lab 3/  
script_rockyou/rockyou_mod.dic" --potfile-disable
```

4.4. Identifica nomenclatura del output

El output de hashcat presenta varias secciones de información relevante. A continuación, se proporciona una descripción detallada de las secciones más importantes:

- **OpenCL API:** Muestra la versión de la API OpenCL utilizada y la plataforma de cálculo (en este caso, Intel Corporation). También informa sobre las capacidades del dispositivo de cálculo, incluyendo la cantidad de memoria disponible.
- **Minimum/Maximum password length:** Indica las longitudes mínima y máxima de la contraseña que puede manejar el kernel de hashcat.
- **Hashes:** Muestra estadísticas sobre los hashes que se van a romper, incluyendo el número total de hashes, los hashes únicos y las sales únicas.
- **Bitmaps:** Detalla la configuración utilizada para la optimización del bitmap.
- **Optimizers applied:** Enumera las optimizaciones aplicadas al proceso de cracking.
- **Dictionary cache hit:** Indica que hashcat ha encontrado y está utilizando una caché del diccionario previamente construida, acelerando así el proceso.
- **Session/Status:** Proporciona el nombre de la sesión y el estado actual del proceso. En este caso, el status es `Cracked`, lo que significa que hashcat ha conseguido romper el hash.
- **Hash.Mode/Hash.Target:** Muestra el modo de hash utilizado y el archivo de destino que contiene el hash a romper.

- **Time.Started/Time.Estimated:** Indica cuándo comenzó el proceso y cuándo se estima que terminará.
- **Speed:** Proporciona detalles sobre la velocidad de cracking, incluyendo el número de hashes por segundo.
- **Recovered:** Muestra cuántos hashes se han descifrado con éxito.
- **Progress:** Indica el progreso del proceso de cracking.
- **Restore.Point:** Muestra el punto de restauración, que es útil si el proceso de cracking necesita ser detenido y reiniciado más tarde.
- **Candidate.Engine/Candidates:** Muestra el motor de generación de candidatos utilizado y los rangos de contraseñas candidatas que se están probando.

4.5. obtiene contraseña con aircrack-ng

En la imagen se puede apreciar la contraseña obtenida usando aircrack-ng

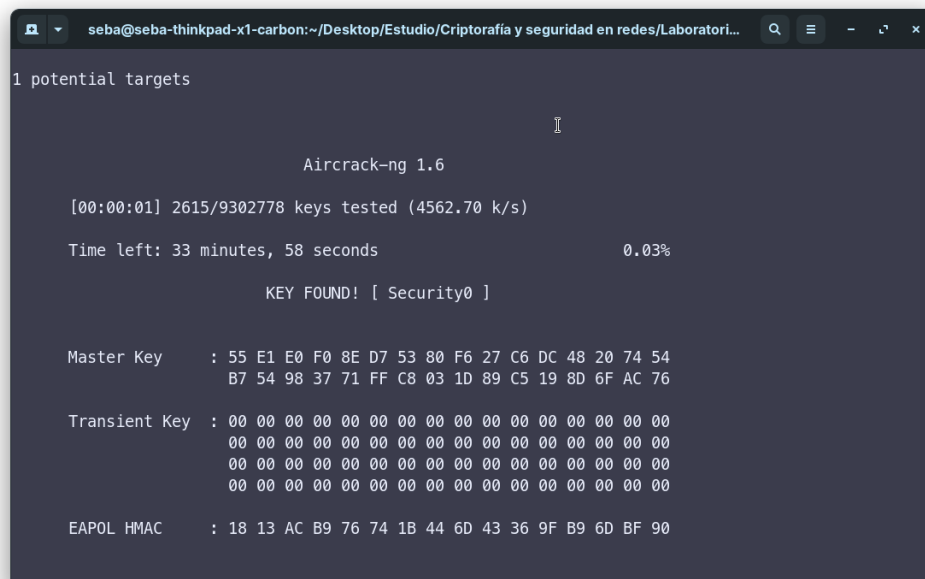


Figura 4: Contraseña usando Aircrack-ng.

4.6. identifica y modifica parámetros solicitados por pycrack

El código con los parámetros modificados:

[illegible]

Figura 5: Parametros pycrack.

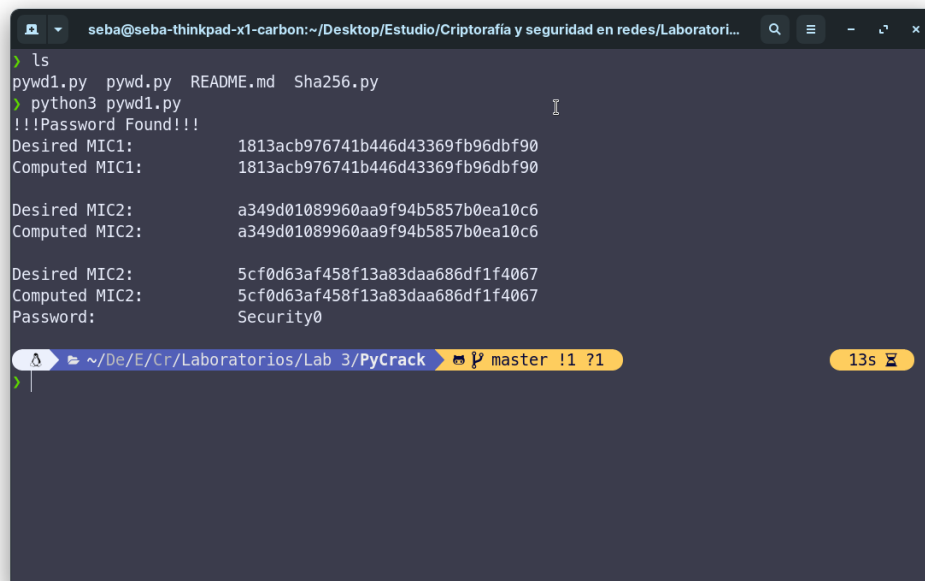
4.7. Descripción de los parámetros del script

El script de Python realiza varias operaciones con el objetivo de ejecutar un ataque de diccionario offline contra un punto de acceso. Los parámetros utilizados en el script son:

- **S**: Esta es una lista que se llena con las contraseñas del archivo de contraseñas.
- **ssid**: Este es el nombre del punto de acceso al que se está intentando acceder.
- **aNonce**: Este es el número de nonce del autenticador, proporcionado en formato hexadecimal y convertido a bytes.
- **sNonce**: Este es el número de nonce de la estación, proporcionado en formato hexadecimal y convertido a bytes.
- **apMac**: Esta es la dirección MAC del punto de acceso (AP), proporcionada en formato hexadecimal y convertida a bytes.
- **cliMac**: Esta es la dirección MAC del cliente, proporcionada en formato hexadecimal y convertida a bytes.
- **mic1, mic2, mic3**: Estos son los valores MIC (Message Integrity Check) para cada uno de los mensajes en el intercambio de handshake.

- **data1, data2, data3:** Estos son los datos completos del marco 802.1x para cada uno de los mensajes en el intercambio de handshake, con el campo MIC establecido en ceros y convertido a bytes.

4.8. obtiene contraseña con pycrack



```
seba@seba-thinkpad-x1-carbon:~/Desktop/Estudio/Criptografía y seguridad en redes/Laboratori...  
> ls  
pywd1.py  pywd.py  README.md  Sha256.py  
> python3 pywd1.py  
!!!Password Found!!!  
Desired MIC1:      1813acb976741b446d43369fb96dbf90  
Computed MIC1:     1813acb976741b446d43369fb96dbf90  
  
Desired MIC2:      a349d01089960aa9f94b5857b0ea10c6  
Computed MIC2:     a349d01089960aa9f94b5857b0ea10c6  
  
Desired MIC2:      5cf0d63af458f13a83daa686df1f4067  
Computed MIC2:     5cf0d63af458f13a83daa686df1f4067  
Password:          Security0  
  
~ /De/E/Cr/Laboratorios/Lab 3/PyCrack  master !1 ?1 13s
```

Figura 6: Contraseña obtenida con pycrack.

Conclusiones y comentarios

En conclusión, durante el desarrollo del laboratorio se lograron obtener contraseñas de redes inalámbricas tanto con cifrado WEP como con cifrado WPA2. Se pudo destacar que el cifrado WEP es obsoleto y fácilmente vulnerable, lo cual lo hace poco seguro y poco recomendable para su uso en redes actuales. Por otro lado, el cifrado WPA2 se mostró más robusto y resistente a ataques, lo que lo convierte en una opción más segura para proteger redes inalámbricas.

Se pudo apreciar que el proceso de obtención de contraseñas requiere la recopilación de una cantidad significativa de paquetes, especialmente en el caso del cifrado WEP, donde se necesitan más de 5000 IVs para llevar a cabo un ataque exitoso. Esto se debe a las debilidades del algoritmo de cifrado y a la necesidad de recopilar suficiente información para realizar un análisis estadístico y descubrir la clave.

En resumen, se evidenció la importancia de utilizar métodos de cifrado seguros y actualizados para proteger las redes inalámbricas, así como la necesidad de implementar contraseñas

robustas y evitar el uso de claves débiles o predecibles. Asimismo, se reafirmó la importancia de mantenerse actualizado sobre las vulnerabilidades y las mejores prácticas de seguridad en redes inalámbricas para garantizar la protección de la información.