Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN TECNOLOXÍAS DA INFORMACIÓN

Herramienta de auditoría de seguridad en redes inalámbricas para pequeñas empresas

Estudiante: Anxo Otero Dans
Dirección: Francisco Javier Nóvoa de Manuel
Dirección: José Carlos Dafonte Vázquez

A Coruña, xuño de 2021.

Lorem Ipsum

Agradecimientos

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Resumen

En la actualidad, el uso de las redes inalámbricas crece exponencialmente en entornos empresariales de todo tipo. Si bien existe una gran cantidad de soluciones adaptadas al ámbito de las grandes organizaciones que permiten realizar tareas de monitorización y auditoría, tanto a nivel físico como de enlace de datos y red, existen muy pocas soluciones en el mercado que se adapten a las necesidades y restricciones económicas de las pequeñas y medianas empresas.

En 2020 en España, el 95% de las empresas son micropymes. Estas organizaciones se caracterizan por no disponer de personal propio de tecnologías de la información y por buscar soluciones de conectividad muy económicas. Además, estos usuarios tienen a su alcance una gran cantidad de productos y servicios tecnológicos que rara vez comprenden y que tienden a funcionar prácticamente durante toda su vida útil con las configuraciones por defecto, lo que entraña un serio peligro que estos usuarios normalmente ignoran.

En este contexto se crea una herramienta que permita realizar auditorías de seguridad de redes inalámbricas en entornos empresariales basados en hardware de bajo coste y código abierto que requiera una mínima intervención y conocimiento por parte del usuario.

Abstract

Nowadays, the use of wireless networks is growing exponentially in business environments of all kinds. Although there is a large number of solutions adapted to large organizations that allow monitoring and auditing tasks, at a physicial level as well as data link and network, there are very few solutions on the market that adapt to the needs and economic constraints of small and medium-sized enterprises.

In 2020 in Spain, 95% of companies are micro-SMEs. These organizations are characterized by not having their own TI staff and by looking for very cheap connectivity solutions. In addition, these users reach a large number of technology products and services that they rarely understand and that work practically for their entire useful life with the default settings, which poses a serious danger that these users usually ignore.

In this context, a tool is created that allows security audits of wireless networks in business environments based on low-cost hardware and open source that requires minimal intervention and knowledge by the user.

Palabras clave:

- First itemtext
- Second itemtext
- Last itemtext
- First itemtext
- Second itemtext
- Last itemtext
- First itemtext

Keywords:

- First itemtext
- Second itemtext
- Last itemtext
- First itemtext
- Second itemtext
- Last itemtext
- First itemtext

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introdución

En este capítulo se comentará la motivación y objetivos del proyecto. Además, se hará una pequeña explicación de la estructura que seguirá la memoria

1.1 Motivación

En la actualidad, el uso de las redes inalámbricas crece exponencialmente en entornos empresariales de todo tipo. Si bien existe una gran cantidad de soluciones adaptadas al ámbito de las grandes organizaciones que permitan realizar tareas de auditoría y monitorización, existen muy pocas soluciones en el mercado que se adapten a las necesidades y restricciones económicas de las pequeñas y medianas empresas.

La gran mayoría de empresas en España son micropymes. Normalmente, estas organizaciones no disponen de personal propio de Tecnologías de la Información (TI) y por ello buscan soluciones de conectividad más económicas. Además, estos usuarios tienen al alcance servicios tecnologícos que rara vez comprenden y que pueden ocasionar un peligro que ignoran.

De la misma manera que el uso de redes inalámbricas crece, también lo hacen los ciberataques. En este último año, debido a la pandemia ocasionada por el SARS-CoV-2 (COVID-19), la mayor parte de la población mundial se ha visto obligada a confinarse en sus casas propiciando un aumento del tráfico de internet, tanto por motivos personales, como por cuestiones laborales. Y con ello, la protección contra ciberataques ha ganado una gran importancia.

Es aquí donde entra en juego la creación de una herramienta que permita auditar redes inalámbricas en entornos principalmente empresariales que ayude a monitorizar las redes y sus dispositivos y detectar posibles brechas de seguridad.

1.2 Objetivos

El objetivo principal del trabajo es desarrollar un sistema basado en placas de bajo coste que realice auditorías de seguridad inalámbricas con una intervención mínima del usuario. Para ello se establecen unos objetivos y funcionalidades básicas de la herramienta:

- **Detección y enumeración de redes inalámbricas:** Haciendo uso de un adaptador de red externa 3.6.1 en modo monitor se analiza el tráfico de red y se detectan las diferentes redes inalámbricas del entorno.
- **Extracción de características de cada red:** Para cada red inalámbrica se muestra una lista de características básicas así como un estudio de la seguridad de la red y de la calidad y potencia que emite su señal.
- **Inventario de dispositivos:** Generación de un inventario de dispositivos de cada Wireless Local Area Network (WLAN), mediante autenticación previa, trantando de identificar sus características.
- **Interfaz Web:** Implementación de una interfaz web sencilla de manera que toda la información obtenida se muestre al usuario de forma inteligible ya que suponemos que éste tendrá conocimientos muy básicos de TI.
- **Servidor central:** Desarrollo de un mecanismo de envío de información a un servidor central por medio de una Application Programming Interfaces (API) Representational State Transfer (REST) de manera que haya un punto centralizado para controlar las diferentes áreas del entorno.

En la imagen 4.1 se muestra un esquema básico de la arquitectura de la herramienta.

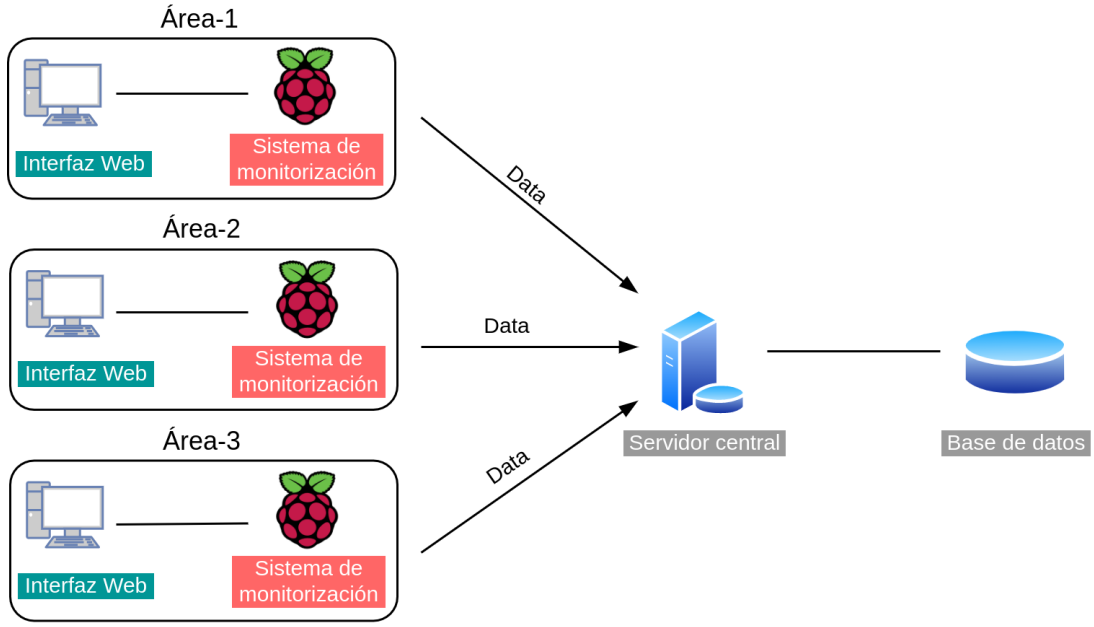

Figura 1.1: Arquitectura básica de la herramienta

1.3 Estado del arte

Existen numerosas herramientas de código abierto para la monitorización y análisis de redes inalámbricas. En este estudio [1] se hace una comparación entre diversas herramientas comunes para el análisis de tráfico de red.

La primera herramienta a la que hace referencia es **Wireshark** [1], una de las herramientas más conocidas para la visualización del tráfico de red. Wireshark permite capturar todo tipo de tráfico en tiempo real a través de una determinada interfaz de red. A pesar de esto, Wireshark tiene bastantes limitaciones, no está pensado para auditorías de seguridad en redes como tal sino que es una simple herramienta para la visualización de paquetes de datos, no dispone de una funcionalidad para poder visualizar y gestionar cada una de las redes inalámbricas de forma individual ni compararlas con otras. Además, la información no se muestra en una forma muy inteligible para un usuario sin conocimientos de TI.

A continuación se habla de **Tcpdump**, una herramienta accesible por línea de comandos muy similar a Wireshark en el sentido en el que dispone de una funcionalidad de captura de tráfico de red pasiva, pero en este caso nos muestra simplemente la información más relevante. Las ventajas de esta herramienta es que la información se muestra de una forma más inteligible para un usuario sin conocimientos de TI, pero sigue sin ser de mucha utilidad ya que no dispone funcionalidades para la gestión de redes ni dispositivos.

[1] https://www.wireshark.org/

Una de las herramientas conocidas que más se aproxima a nuestra funcionalidad principal es **airodump-ng** perteneciente a la suite de herramientas de seguridad **aircrack-ng** [2]. Airodump-ng es una herramienta accesible por línea de comandos que a diferencia de las anteriores, no se centra en la visualización de tráfico de red sino que se encarga de filtrar los datos capturados y mostrar de forma inteligible un inventario dinámico de redes inalámbricas con sus características más relevantes. Pero de la misma manera que las anteriores, tiene algunas limitaciones. Como habíamos comentado, airodump-ng se centra en mostrarnos las redes inalámbricas y las características más relevantes pero no nos permite ver más allá de eso, no podemos visualizar el resto de información de una red inalámbrica ni podememos gestionarla de ninguna manera sino que necesitariamos herramientas adicionales. Tampoco sirve como sistema de detección ya que no nos permite visualizar paquetería sensible como la comentada en el apartado 2.2.

Otro de los puntos clave de este proyecto es la utilización de placas base como sistema de monitorización independiente. En este estudio [2] se muestra la creación de una **herramienta de monitorización** de redes inalámbricas haciendo uso de placas base **Arduino**. Si bien es cierto que este estudio hace una aproximación del núcleo de nuestro proyecto en cuanto al uso de componentes *hardware*, la herramienta carece de complejidad ya que apenas muestra las redes inalámbricas disponibles sin enseñar sus características y, de la misma forma que las herramientas anteriores, es una herramienta de simple visualización sin capacidad de extracción de conclusiones ni de gestión de la seguridad.

Estas últimas herramientas tienen en común que sólo nos permiten visualizar tráfico de red fuera de nuestra WLAN, lo que nos limita a la hora de visualizar información sobre los dispositivos conectados a nuestra red local.

Nmap [3] es una de las herramientas más conocidas y más utilizadas de código abierto para el escaneo y auditoría de redes, ya que dispone de una gran cantidad de funcionalidades. Nmap es capaz de realizar un escaneo tanto pasivo como activo de la red y mostrarnos los dispositivos conectados a ésta, además de extraer una gran cantidad de información sobre estos dispositivos. Es una herramienta de bastante utilidad pero de la misma manera que las anteriores no dispone de una interfaz gráfica y en este caso, solo permite el descubrimiento de dispositivos de una red local y no de otras redes Wi-Fi como venía haciendo la herramienta airodump-ng.

La conclusión que sacamos es que, a pesar existir una gran cantidad de herramientas muy útiles como las comentadas anteriormente, no existe una herramienta de código abierto que contenga todas las funcionalidades en una, sino que se necesitaría complementar múltiples herramientas para poder conseguir el sistema final que se tiene como objetivo en este trabajo.

[2] https://www.aircrack-ng.org/
[3] https://nmap.org/

1.4 Estructura de la memoria

La memoria seguirá la siguiente estructura:

1. **Introducción:** En este aparado se habla de la motivación de la que surge este proyecto a raíz de las necesidades actuales y se definen unos objetivos a alcanzar. También se habla de las herramientas y trabajos ya existentes en el ámbito de auditoría de redes y se expone la estructura que va a tener la memoria.

2. **Conceptos previos:** Se detallan conceptos relacionados con la auditoría de redes. Concretamente, se explica la importancia de la clasificación de paquetería de datos y se detallan algunos de los ataques a redes inalámbricas más comunes.

3. **Fundamentos tecnológicos:** En este apartado se exponen las tecnologías utilizadas y se justifica su uso para el desarrollo del proyecto.

4. **Metodología y gestión del proyecto:** Se habla de la metodología utilizada en el desarrollo del proyecto, la planificación de éste y la estimación de recursos y costes.

5. **Implementación:** Se detalla el análisis, diseño e implementación de cada uno de los componentes del sistema.

6. **Pruebas y resultados:** En este apartado se exponen los resultados de las pruebas realizadas con la herramienta en un entorno real.

7. **Conclusiones:** Se hace un análisis de los resultados obtenidos y se habla del posible trabajo futuro a realizar sobre el proyecto.

Capítulo 2

Conceptos Previos

En este capítulo hablaremos sobre conceptos referentes al proyecto que ayudarán a comprender de forma más sencilla la funcionalidad de la herramienta y su diseño.

2.1 Visualización del tráfico de red

Cuando un dispositivo envía datos por la red lo hace en forma de tramas, estas tramas son piezas de información que contienen datos binarios en crudo. Es aquí donde entran en juego los llamados *packet sniffer*, que son herramientas que se encargan de capturar esas tramas y de mostrar la información en un formato legible al usuario.

Normalmente, las tarjetas de red de nuestros dispositivos personales tienen como función el conectarse a un Wireless Access Point (AP) de una red inalámbrica y el único tráfico de red que podemos capturar o visualizar es el generado por los dispositivos de la red local a la que estamos conectados.

En cambio, existe una opción en las tarjetas de red que nos permite visualizar el tráfico de red de todas las redes inalámbricas de nuestro entorno. Para ello, necesitamos tener activada la opción de modo monitor en nuestra interfaz de red inalámbrica, si bien es cierto que muchos de los portátiles modernos disponen de esta funcionalidad, lo más probable es que las capacidades de la tarjeta de red inalámbrica no sean suficientes para poder capturar todo el tráfico y extraer todas las características de una red. Para ello necesitamos hacer uso de una Wireless Network Interface Controller (WNIC) externa.

Existen multitud de WNIC en forma de antena en el mercado que son fáciles de integrar a nuestro dispositivo a través de un puerto Universal Serial Bus (USB) y una instalación de *drivers* sencilla.

Como bien se ha comentado, no todas las WNIC tienen las mismas capacidades ya que dependiendo de la marca y modelo pueden variar en el alcance de la señal, en el tipo de frecuencia en la que trabaja y en los IEEE 802.11 estándares soportados.

2.2 Extracción de información según el tipo de tráfico

La clasificación del tráfico de red tiene un papel muy importante en este trabajo ya que dependiendo del tipo de trama que se encuentre, se puede extraer un tipo de información u otra.

Existen cuatro tipos de tramas en el estándar IEEE 802.11 [3]: de Gestión, de Control, de Datos y un último tipo para tramas de extensión. Las tramas de Gestión son las encargadas de controlar y gestionar las comunicaciones entre el AP y los dispositivos por lo que tienen un papel clave en la monitorización y gestión de las redes. A continuación haremos una breve descripción de algunas de las más importantes:

- ***Beacon Frames:*** Es el tipo de trama que más abunda ya que es transmitida por el propio AP periódicamente para anunciar la presencia de la red WLAN. Contiene toda la información de la red por lo que juega un papel muy importante en la detección de redes y en la extracción de sus características.
- ***Deauthentication Frames:*** El AP envía una notificación a los dispositivos conectados a la red avisando de que la conexión se va a terminar, en caso de que el dispositivo no esté conectado, impide que éste se conecte. Es potencialmente peligroso para posibles ataques que comentaremos más adelante.
- ***Dissasocitation Frames:*** De la misma manera que el anterior, fuerza a un dispositivo a desconectarse de la red. En este caso, para que ocurra tiene que haber una asociación previa y para que haya una asociación tiene que haber una autenticación.
- ***Authentication Frames:*** Una vez un dispositivo descubre una red e intenta conectarse a ella, éste envía una trama de autenticación y comienza un proceso de asociación.
- ***Association Request/Response Frames:*** Cuando el proceso de autenticación se inicia, el dispositivo envía una petición de asociación a la estación correspondiente y ésta le responde para completar así el proceso de autenticación/asociación a la red. En la figura 2.1 podemos ver el proceso completo.

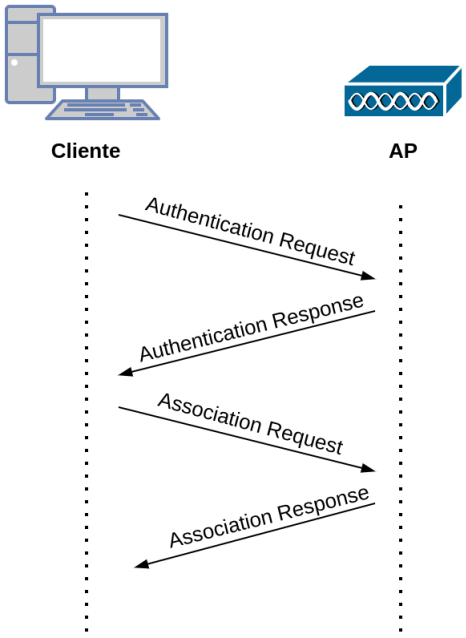

Figura 2.1: Proceso de autenticación de un dispositivo a la red

2.3 Clasificación de ataques en redes inalámbricas

2.3.1 Ataque de Desautenticación

Es un tipo de ataque de Denial of Service (DoS) que bloquea la comunicación entre un dispositivo y un AP. Este ataque se aprovecha de las tramas de desautenticación que hemos comentado en el apartado 2.2.

Un atacante malicioso que quiera perjudicar la seguridad e integridad de nuestra red puede spoofear la dirección Media Access Control (MAC) de los dispositivos de la red enviando tramas de desautenticación al AP para desconectarlos. Existen dos tipos de ataque de desautenticación:

- **Ataque dirigido:** El atacante elige a uno de los dispositivos de la red como víctima y spoofea su dirección MAC envíando *deauthentication frames* al AP para desconectarlo individualmente de la red.

- **Ataque global:** El atacante genera *deauthentication frames* con dirección Broadcast de manera que el AP desconecta a todos los dispositivos de la red.

Ya que las tramas de desautenticación son notificaciones, no pueden ser evitadas y es por eso que este ataque supone un gran peligro para nuestra red.

Uno de los detalles más importantes de este ataque es que, si el atacante está monitorizando la red en el momento de desautenticar a un cliente, éste al volver a conectarse genera

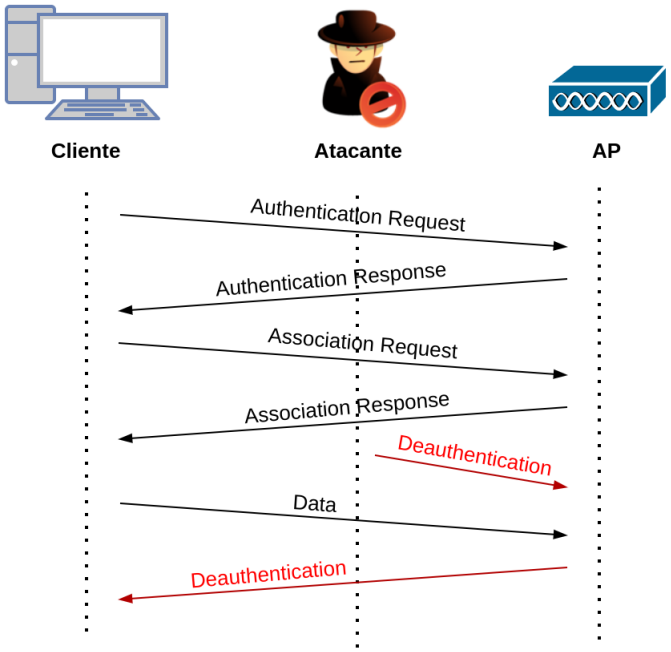

Figura 2.2: Proceso de un ataque de desautenticación

un Wi-Fi Protected Access (WPA) Handshake que puede ser capturado por el atacante y posteriormente usarlo para descifrar la contraseña del AP mediante técnicas de fuerza bruta.

Existen múltiples herramientas para poder realizar este tipo de ataque, una de las más conocidas es **aireplay-ng** utilizando el parámetro –deauth.

En la figura 2.2 podemos ver el proceso de desautenticación de un cliente.

2.3.2 Ataque de Desasociación

Este ataque es muy similar al ataque de desautenticación 2.3.1, de la misma manera que el anterior, un atacante spoofea la dirección MAC de un dispositivo y fuerza a desconectarlo enviando *dissasociation frames*.

A diferencia del ataque de desautenticación, para que un dispositivo se desasocie éste tiene que estar autenticado y asociado a un AP previamente.

Normalmente estos dos ataques van de la mano ya que con el ataque de desasociación puedes expulsar a los dispositivos conectados a una red inalámbrica y con el ataque de desautenticación puedes evitar que éstos se vuelvan a conectar.

Una herramienta muy útil para éste tipo de ataques es **mdk3**, que con el parámetro *d* se pueden realizar ataques de desautenticación y desasociación de forma dirigida utilizando una *blacklist* de puntos de acceso Wi-Fi, también existe la posibilidad de atacar de forma global a todos los puntos de acceso en rango al no especificar ningún AP como víctima.

2.3.3 Ataque de Falsa Autenticación

Normalmente es mucho más fácil atacar a una red con dispositivos conectados, en caso de que una red no disponga de dispositivos limita los vectores de ataque de un atacante. Es aquí donde entra en juego el ataque de falsa autenticación que consiste en, tratar de autenticar y asociar clientes falsos a una red.

A pesar de ser capaces de asociar un cliente falso a una red, éste no va a generar ningún WPA Handshake ya que no tiene conocimiento de la contraseña.

Por ello, uno de los posibles usos de éste tipo de ataque es el **ataque masivo de autenticación**, que consiste en autenticar múltiples dispositivos a la red de forma masiva para saturarla impidiendo así a los dispositivos reales conectarse a ella.

De la misma manera que los ataques mencionados en los apartados 2.3.1 y 2.3.2, las herramientas **mdk3** y **aireplay-ng** también permiten realizar este tipo de ataque.

2.3.4 Ataque de Beacon Flood

Como comentamos en el apartado de tráfico de red 2.2, los *beacon frames* contienen toda la información de un AP y se transmiten en claro por lo que son los encargados de anunciar que redes hay en nuestro entorno y cuáles son sus características.

Este ataque se encarga de hacer exactamente eso, inundar la red de forma masiva con *beacon frames* anunciando puntos de acceso falsos de manera que saturan la red e imposibilitan a los dispositivos visualizar y conectarse a las redes Wi-Fi reales. En la figura 2.3 podemos ver una demostración del ataque.

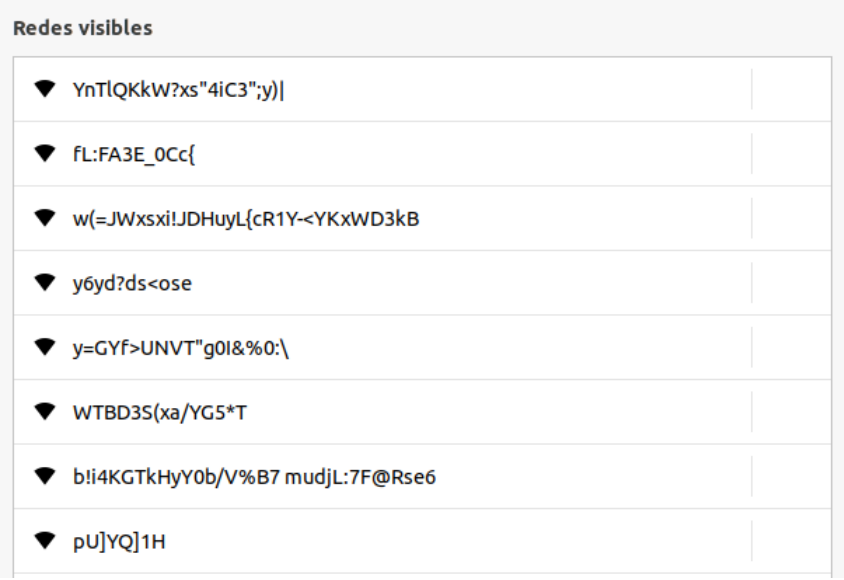

Figura 2.3: Demostración del ataque Beacon Flood

2.3.5 Ataque de Rogue AP

Un ataque de Rogue AP consiste en instalar un AP en una red segura sin autorización previa del administrador.

Este ataque supone un gran riesgo de seguridad para la red interna ya que proporciona una puerta trasera de acceso a usuarios no autorizados. Algunos de los riesgos potenciales son:

- Permitir a un atacante realizar un ataque Man-in-the-Middle (MitM) estableciendo conexiones individuales con las víctimas e intercambiando mensajes con ellas haciéndoles creer que es un dispositivo autorizado.
- Inundar la red con tráfico provocando así un DoS.
- Enviar notificaciones atractivas de falso SSID tales como conexión a internet abierta tratando de redirigir al usuario a un portal cautivo para poder robarle información sensible. Uno de los ataques más conocidos de Rogue AP es el Evil twin.

2.4 Características básicas de las redes inalámbricas

Son muchas las características que componen una red inalámbrica y éstas van a ser clave para la extracción de cierta información como la potencia y calidad de la señal que emite el AP o cuánto de segura es una red. Las características más básicas son:

- **SSID:** El Service Set Identifier (SSID) es básicamente el nombre que que se le da a cada una de las redes. Debido a que múltiples redes pueden coexistir en un mismo espacio áereo, se necesita un identificador para diferenciar cada uno de los dispositivos de una misma red, a pesar de esto, existe la posibilidad de que múltiples redes adopten el mismo SSID.
- **BSSID:** El Basic Service Set Identifier (BSSID) es, de de la misma manera que el anterior, un identificador para cada uno de los puntos de acceso de una red. Se forma con la dirección MAC del dispositivo por lo que en este caso si se trata de un identificador único.
- **RSSI:** El Received Signal Strength Indicator (RSSI) es el indicador de fuerza de la señal recibida de una red. Es una escala de referencia en relación a 1mW y normalmente se expresa en Decibelio-milivatio (dBm). Éste tiende a 0 y se expresa en números negativos, por lo que cuánto mayor esté del 0, mayor será la potencia de la señal recibida.

BSSID	PWR	Beacons	#Data,	#/s	CH	MB	ENC	CIPHER	AUTH	ESSID
0E:41:58:00:82:BD	-66	183	125	0	11	130	WPA2	CCMP	PSK	MiFibra-58BA
E4:C3:2A:7E:32:9D	-68	4	0	0	7	130	WPA2	CCMP	PSK	Richard_EXT
04:A2:22:6F:58:BC	-72	161	45	0	11	130	WPA2	CCMP	PSK	MiFibra-58BA
F8:8E:85:33:B9:FA	-74	1	0	0	1	130	WPA2	CCMP	PSK	MOVISTAR_B9F9
0A:4A:77:E8:74:05	-76	2	0	0	36	1733	WPA2	CCMP	PSK	MIWIFI_2G_6uef
38:70:0C:35:10:B0	-76	1	0	0	8	130	WPA2	CCMP	PSK	R-WLAN 2A

Figura 2.4: Características básicas de una red capturadas por la herramienta airodump-ng

- **Canal:** El canal de una red inalámbrica indica la radiofrecuencia en la que ésta se encuentra. Dependiendo del tipo de red e incluso del país, cada red puede trabajar en una frecuencia distinta. Las redes domésticas y las más habituales suelen trabajar en la banda de frecuencia 2.4GHz que se compone de los canales del 1 al 14, también están las redes que trabajan en la banda de frecuencia 5GHz que son las más actuales y disponen de una gran cantidad de canales por lo que la saturación y colapso de redes es menor.

  A pesar de esto, las redes no se encuentran únicamente en una misma frecuencia y en un mismo canal de manera estática sino que dependiendo de la configuración que tengan pueden ir haciendo channel hopping automáticamente buscando una menor saturación de la red y un mayor ancho de banda.

- **Seguridad:** Hay varias configuraciones de seguridad que puede tener una red wifi. Son cuatro los protocolos básicos de seguridad inalámbrica que existen hasta el momento:

  - **WEP:** Proporciona una baja seguridad y está considerado como la configuración más vulnerable de todas (después de las redes abiertas). No se recomienda su uso.
  - **WPA:** Fue creado como un protocolo temporal de migración para solucionar las vulnerabilidades del protocolo WEP. Debido a que fue creado para cubrir una necesidad temporal, éste dispone de mejoras de seguridad pero continua siendo muy vulnerable a ataques.
  - **WPA2:** Este protocolo es la estandarización del protocolo WPA y constituye una mejora de seguridad muy importante para las redes inalámbricas.
  - **WPA3:** Es el protocolo de seguridad más avanzado hasta el momento, protege a la red de una gran cantidad de ataques y limita sus vulnerabilidades respecto a los protocolos anteriores.

  Cada uno de los protocolos va acompañado de más elementos de seguridad como los algoritmos de cifrado y de autenticación.

Capítulo 3

Fundamentos tecnológicos

En este capítulo se explicarán las tecnologías usadas y se justificará la elección de cada una de ellas. Se dividirán en múltiples bloques dependiendo de su papel dentro del desarrollo del proyecto.

3.1 Desarrollo de la aplicación principal

En esta sección se detallarán las tecnologías empleadas en el desarollo del bloque principal de la herramienta.

3.1.1 Python

Python[1] [4] es un lenguaje de programación interpretado de propósito general. Es multiparadigma, lo que quiere decir que soporta diferentes paradigmas de programación como: programación orientada a objetos, programación imperativa y hasta programación funcional.

La elección de Python como lenguaje principal del proyecto es debido principalmente al uso de la la librería Scapy 3.1.2, a su gran versatilidad y a su facilidad para desarrollar de una manera ágil y rápida.

3.1.2 Scapy

Scapy [2] [5] es una potente librería de Python que permite la manipulación de paquetes de datos de red.

Es capaz de falsificar o decodificar paquetes de una gran cantidad de protocolos, enviarlos por cable, capturarlos, emparejar solicitudes y respuestas y mucho más. Con todas las funcionalidades de las que dispone es capaz de realizar la mayoría de tareas clásicas como *scanning*, *tracerouting*, *probing*, *network discovery* e incluso es capaz de realizar ataques.

[1] https://www.python.org/
[2] https://scapy.net/

La elección de esta tecnología se debe a la capacidad que tiene de sustituir las herramientas más clásicas y conocidas de manipulación de tráfico de red de forma que podamos tener incluso un mayor manejo de las utilidades que éstas proporcionan.

3.1.3 Django

Django [3] es un *framework* de desarrollo web basado en Python. Django facilita en gran medida el desarrollo de la aplicación proporcionando una gran cantidad de funcionalidades internas tales como: la creación de un panel de administración, la creación y autenticación de usuarios dentro de la aplicación y mucho más.

Django sigue un patrón de diseño Model View Template (MVT) a diferencia del típico Model View Controller (MVC), que básicamente permite desarrollar la aplicación de manera modular y eficiente.

La elección de éste *framework* es debido a que está basado en Python y al conocimiento previo por parte del alumno.

3.1.4 Celery

Celery [4] es un programa de código abierto que permite la implementación de una cola de tareas de forma asíncrona basada en el paso de mensajes distribuidos. Celery también permite la planificación de tareas pero su labor principal es la de realizar operaciones en tiempo real. Su uso dentro de la aplicación se debe a la posibilidad de ejecución de tareas de largo periodo de tiempo sin interferir en la comunicación con el usuario.

También dispone de una fácil integración con Django y Python mediante la instalación a través de The Python Package Index (PyPI)

3.2 Desarrollo de la interfaz web

En esta sección hablaremos sobre las tecnologías empleadas para el desarollo de la interfaz web de la aplicación.

3.2.1 HTML y CSS

HyperText Markup Language (HTML) es el estándar de lenguaje de marcado para páginas web. Los buscadores web reciben documentos HTML y los renderizan en páginas web multimedia.

Cascading Style Sheets (CSS) es un lenguaje de diseño gráfico que se encarga de definir y crear la presentación de un documento estructurado escrito en HTML.

[3] https://www.djangoproject.com/
[4] https://docs.celeryproject.org/

3.2.2 JavaScript

JavaScript [5] es un lenguaje de programación multiparadigma. Se implementa normalmente como parte de un navegador web incluyendo mejoras en la interfaz de usuario y páginas web dinámicas.

JavaScript contiene una gran cantidad de librerías que permiten incluir múltiples funcionalidades a nuestro sitio web. En nuestro caso se han utilizado tres librerías adicionales para la creación del proyecto: JQuery, Vis.js y Chart.js

JQuery

JQuery [6] es una librería multiplataforma de JavaScript que permite: simplificar la manera de interactuar con los documentos HTML, manipular el árbol Document Object Model (DOM), manejar eventos, desarrollar animaciones y agregar interacción con la técnica Asynchronous JavaScript And XML (AJAX) a páginas web.

Vis.js

Vis.js [7] [6] es una librería de visualización dinámica basada en navegador. Ha sido diseñada para manejar una gran cantidad de datos dinámicos y manipularlos de manera que podamos verlos de una forma más atractiva.

La librería consiste en varios componentes de visualización gráfica, pero en nuestro caso sólo usaremos el componente *Network* para la creación de un mapa dinámico del entorno Wi-Fi.

Chart.js

Chart.js [8] es una librería de JavaScript de código abierto para la visualización de datos en forma de gráficos.

La elección de esta librería se debe principalmente a que es de código abierto y a la facilidad de aprendizaje gracias a su documentación.

3.3 Persistencia

En este apartado hablaremos de las distintas soluciones de almacenamiento que se han utilizado en el desarrollo del proyecto.

[5] https://www.javascript.com/
[6] https://jquery.com/
[7] https://visjs.org/
[8] https://www.chartjs.org/

3.3.1 PostgreSQL

PostgreSQL [9] es un sistema de base de datos relacional, orientado a objetos y de código abierto.

En este caso se ha empleado PostgreSQL como sistema de almacenamiento de datos de la aplicación principal ya que se requiere de un gran manejo de los campos de cada uno de modelos almacenados.

La elección de PostgreSQL se debe a su gran compatibilidad con Django ya que, no sólo es el sistema de almacenamiento que recomienda la propia plataforma, sino que también proporciona ventajas con respecto a otros sistemas soportados por Django.

3.3.2 MongoDB

MongoDB [10] es sistema de base de datos no relacional, orientado a documentos y de código abierto.

El hecho de ser un sistema no relacional permite guardar los elementos como objetos en vez de la clásica estructura de tablas. La ventaja que tiene esto es la rapidez y facilidad que tiene de almacenar datos de todo tipo y de manera escalable.

Se ha utilizado MongoDB como base de datos para la implementación del servidor central ya que éste recibirá una gran cantidad de datos en formato JSON que es el formato soportado por MongoDB y de esta manera no hay necesidad de modificar el formato de los datos recibidos para poder almacenarlos.

3.4 Despliegue

En esta sección se hablará sobre las tecnologías que se han utilizado para el despliegue de la aplicación.

3.4.1 Docker

Docker [11] es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores *software*, proporcionando así una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.

También se ha utilizado la herramienta Docker-Compose que sirve para el despliegue de múltiples contenedores a través de un fichero YAML Ain't Markup Language (YAML) que configura los servicios de la aplicación.

[9] https://www.postgresql.org/
[10] https://www.mongodb.com/
[11] https://www.docker.com/

Esta tecnología tiene un papel clave ya que la finalidad del proyecto es desplegar la aplicación en múltiples dispositivos, por lo que simplifica mucho la tarea de instalación en cada uno de los dispositivos.

3.4.2 Nginx y Gunicorn

Nginx [12] es un servidor web ligero y de alto rendimiento. Las ventajas que tiene Nginx es que es de código abierto y es soportado por todo tipo de sistemas Unix y Windows.

Se ha utilizado Nginx en la parte del servidor central ya que, a pesar de que Django proporciona un servidor propio, éste no es óptimo para su despliegue en producción y se recomienda el uso de un servidor con mayor rendimiento.

Gunicorn [13] es un Web Server Gateway Interface (WSGI), un envoltorio de aplicaciones web para Python que ofrece un punto de entrada y salida para las conexiones. Junto con Nginx compone un combo perfecto para el despliegue de aplicaciones con Django 3.1.3.

3.5 Soporte del desarrollo

En esta sección se hablará sobre las diferentes herramientas adicionales que, si bien no participan en el desarrollo principal de la herramienta, han servido como apoyo para el mismo.

3.5.1 PyCharm

PyCharm [14] es un entorno de desarrollo integrado para Python pertenenciente al grupo de herramientas para desarrolladores JetBrains [15]. Facilita mucho el desarrollo en *frameworks* de Python como Django.

Existen dos opciones: una versión *community*, que es gratuita y una versión *professional*, que es de pago. Ésta última se puede adquirir de manera totalmente gratuita siendo estudiante.

3.5.2 Git

Git [16] es un *software* de control de versiones de código abierto. Permite gestionar proyectos de todo tipo de manera muy sencilla y eficiente.

[12] https://www.nginx.com/
[13] https://gunicorn.org/
[14] https://www.jetbrains.com/es-es/pycharm/
[15] https://www.jetbrains.com/
[16] https://git-scm.com/

3.5.3 GitHub

GitHub [17] es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git.

3.6 Tecnologías Hardware

En esta sección se explicaran los diferentes componentes *hardware* necesarios para el despliegue de la herramienta.

3.6.1 Adaptador de red externo

Para el correcto y completo funcionamiento de la aplicación es necesario un adaptador de red externo que cumpla con los requisitos necesarios.

En este proyecto se ha utilizado el adaptador de red *«CSL-AC1200 USB 3.0 Dual Band WLAN Stick»* que contiene un chip *«Realtek RTL8812AU»* que tiene la capacidad suficiente de reconocer las redes inalámbricas del entorno y de extraer las características necesarias de éstas.

3.6.2 Raspberry Pi

Una Rasperry Pi [18] es una serie de ordenadores de placa simple de bajo coste.

El objetivo de la Raspberry Pi es la de funcionar como un ordenador de pequeño tamaño y de precio reducido. Tiene la ventaja de que se puede personalizar añadiendo diferentes módulos y accesorios para poder realizar multitud de proyectos.

En el proyecto se ha utilizado la *«Raspberry Pi 3 Model B+»*, que junto al adaptador de red externo 3.6.1 componen el sistema de captura principal de la herramienta.

3.6.3 Servidor

Se necesita una infraestructura que sea capaz de correr el servidor central de la herramienta y almacenar su base de datos.

No existen unos requisitos fijos para esta infraestructura ya que dependerá de factores como del tiempo de funcionamiento y del volumen de datos que se genere.

[17] https://github.com/
[18] https://www.raspberrypi.org/

Capítulo 4

Metodología y gestión del proyecto

En este capítulo se describirá y se justificará la metodología de desarrollo empleada en el proyecto. La elección de una metodología se basa en el nivel de amoldamiento en base a las características y los beneficios que obtendríamos respecto a otras.

Se empleará una metodología incremental, mejorando el sistema y añadiéndole nuevas funcionalidades en cada iteración, hasta obtener así el sistema final. Para ello se ha decidido trabajar con base en el marco de desarrollo *Scrum*, adaptándolo a las características de un Trabajo de Fin de Grado.

También hablaremos sobre el resto de aspectos que conforman el proceso de ingeniería tales como: la estimación y gestión de recursos y costes y de la planificación del proyecto.

4.1 Elección de la metodología

La elección de *Scrum* como metodología a seguir en el proyecto se debe a un resultado del análisis y evaluación de las metodologías de desarrollo más utilizadas en la actualidad. Para ello se han establecido unos criterios para la toma de decisión:

- Facilidad de **adaptación** en caso de tener que añadir nuevas funcionalidades en base a las necesidades cambiantes.
- ***Feedback*** del cliente de forma reiterada de manera que se le hagan entregas parciales del producto para ir ajustando y adecuando las necesidades pertinentes.
- Necesidad de una **rápida detección de problemas** para poder solucionarlos fácilmente en siguientes iteraciones y no tener que esperar a la finalización del proyecto.
- **Ajuste proporcional a los conocimientos del desarrollador**. En este caso el estudiante no tenía total conocimiento de las tecnologías que se iban a utilizar en el proyecto por lo que se necesitaba de una metodología que se fuera adaptando al aprendizaje de éstas.

Como se puede observar, los criterios mencionados anteriormente coinciden con los de una metodología ágil y como ya veníamos comentando, se ha elegido la metodología *Scrum*.

4.2 Scrum

Scrum [7] es un marco de trabajo para el desarrollo ágil de *software* que fue identificado y definido por Ikujiro Nonaka y Takeuchi a principios de los 80 al analizar cómo desarrollaban los nuevos productos las principales empresas de manufactura tecnológica. En su estudio, Nonaka y Takeuchi compararon la nueva forma de trabajo en equipo con el avance en formación *scrum* de los jugadores de Rugby. A raíz de esto quedó acuñado el término *Scrum* para referirse a esa nueva forma de trabajo en equipo.

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En *Scrum* se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, *Scrum* está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultado pronto, donde los requisitos son cambiantes y donde la flexibilidad y la productividad son fundamentales.

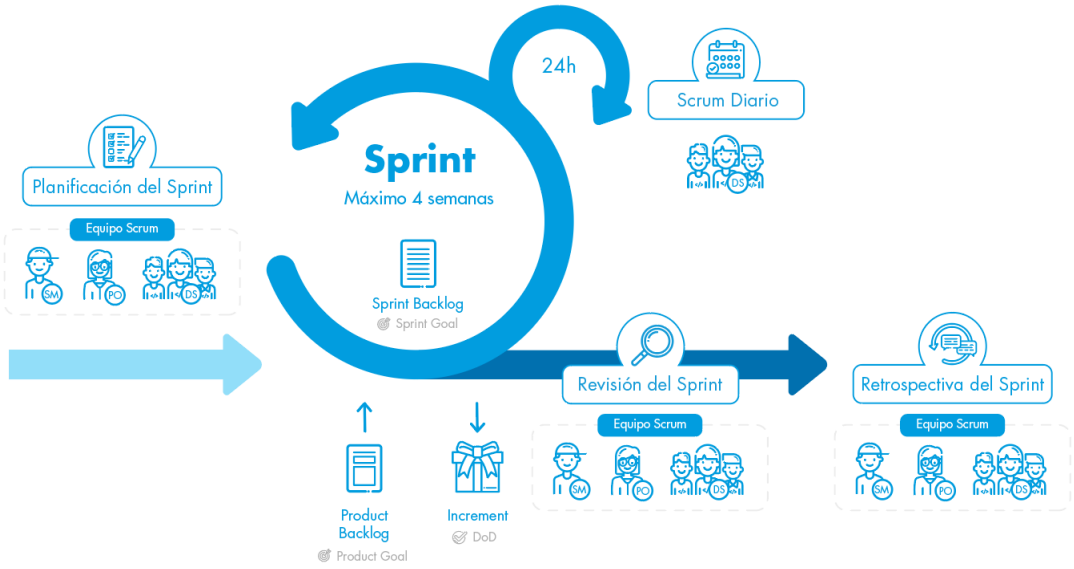

Figura 4.1: Esquema de la metodología Scrum

4.2.1 Roles

La metodología *Scrum* define tres roles indispensables para el correcto desarrollo de un proyecto. Estos roles son: *Product Owner*, *Scrum Master* y *Development Team*.

La funcionalidad de los roles en un equipo *Scrum* es la de poder auto-organizarse de manera que no esté dirigido por alguien externo, es el propio equipo de *Scrum* el que elige la mejor forma de llevar acabo el proyecto.

- ***Product Owner*:** El *Product Owner* es el encargado de maximizar el valor del producto y del trabjo del Equipo de Desarrollo. Están centrados en entender los requisitos empresariales, de los clientes y del mercado, para luego priorizar el trabajo que el equipo debe realizar para cumplirlos. Algunas de las tareas que realizan los *Product Owner* son:

  - Crear y gestionar el *Product Backlog*.
  - Aportan las directrices al equipo sobre qué funcionalidades se entregan en cada incremento.
  - Deciden cuándo lanzar el producto con predisposición hacia una entrega más frecuente.
  - Se asocian estrechamente con el negocio y el equipo para asegurarse de que todos entienden los elementos de trabajo en el *Product Backlog*.

  El *Product Owner* no siempre es el gestor de proyectos, y es importante que sea una única persona ya que los Equipos de Desarrollo no desean directrices cruzadas de diferentes *Product Owner*.

- ***Scrum Master*:** Los *Scrum* Master son los mayores especialistas de *Scrum* en el equipo. Es la persona encargada de representar al equipo frente a la dirección y a los usuarios externos, es decir, trabaja como mediador constante para garantizar la correcta comunicación entre los agentes involucrados.

  Un *Scrum Master* conoce profundamente el trabajo que realiza el equipo y puede ayudarlo a optimizar su transparencia y flujo de entrega. Por un lado, presta servicio al *Product Owner* en la gestión del *Product Backlog* y por otro, presta servicio al Equipo de Desarrollo organizando sus funcionalidades.

- ***Development Team*:** El Equipo de Desarrollo está constituido por varios profesionales (normalmente entre cuatro y nueve personas) con distintas habilidades que desempeñan la función de entregar **incrementos** del producto ya finalizado que se pueda poner en producción al final de cada *sprint*.

Las características principales de un Equipo de Desarrollo son: su multifuncionalidad, la división de responsabilidades y su auto-organización.

En cada *sprint*, el Equipo de Desarrollo construye y entrega un incremento del producto. Cada incremento es un subconjunto del producto, reconocible y visualmente mejorado, que cumple con el criterio de aceptación y está construido con un nivel de calidad llamado «definición de terminado».

4.2.2 Eventos

Los Eventos son protocolos o reuniones realizadas de forma periódica por los equipos de *Scrum*. Todos los eventos tienen una duración máxima que habrá que respetar. Existen cinco eventos en total:

- ***Sprint*:** El *sprint* es el corazón de *Scrum*. Es un **periodo de tiempo** real en el que el equipo de *Scrum* trabaja de forma conjunta para finalizar un **incremento**. La duración de un *sprint* suele oscilar entre las dos y las cuatro semanas.

  La duración de un *sprint* se establece previamente y no se debe cambiar una vez éste ya haya comenzado, de hecho es recomendable que su duración sea igual a lo largo de la duración de todo el proyecto, de esta manera se obtendrán estimaciones más precisas.

  Los *sprints* **constan de un objetivo** y un plan previamente definido que se deberá acatar sin variaciones para el resultado exitoso del producto.

- ***Panificación del Sprint*:** Los *sprints* se planifican a través de una reunión en la que todo el Equipo de Desarrollo planifica el trabajo que se va a realizar durante el *sprint* actual.

  La reunión **la dirige el *Scrum Master*** y, en ella, el equipo decide el objetivo del *sprint*.

- ***Daily Scrum*:** Se trata de una reunión diaria de corta duración (aproximadamente quince minutos) que tiene lugar siempre a la misma hora y en el mismo lugar. Su objetivo es el de sincronizar a todos los miembros del Equipo de Desarrollo para que trabajen en sintonía y se adecúen al objetivo de manera conjunta.

- ***Sprint Review*:** Es una reunión que se realiza al final de cada Sprint, en ella el Equipo de Desarrollo muestra el *Product Backlog* con los elementos finalizados para poder recibir *feedback* y realizar los cambios pertinentes si éstos fueran necesarios.

- ***Sprint Retrospective*:** Es una reunión en la que el Equipo de Desarrollo se reúne para documentar y analizar qué ha funcionado y qué no ha funcionado en el *sprint*, en el proyecto en general, en las personas o relaciones o incluso en las herramientas. La

finalidad es crear un ambiente donde el equipo pueda centrarse en las cosas positivas y en lo que debe mejorarse para la próxima vez.

4.2.3 Artefactos

Los artefactos son todos los elementos que garantizan la transparencia y el registro de la información fundamental del proceso de *Scrum*. Dicho de otra manera, son los recursos que cimientan la **productividad y la calidad** de cualquier proyecto. Estos son:

- ***Product Backlog*:** Es una lista ordenada con todos los requisitos que tiene que cumplir un producto para satisfacer a las necesidades de los clientes potenciales y es la única fuente de requisitos para realizar modificaciones en él. Se trata de una lista dinámica de funciones, requisitos mejores y correcciones que actúa como la entrada para el *Sprint Backlog*.

  Por definición, esta lista nunca está completa ya que cambia a medida que lo hace su entorno y el propio producto para ser apropiado, útil y competitivo. Los ítems incluidos en el *Product Backlog* tienen los siguientes atributos:

  - Descripción
  - Ordenación
  - Estimación
  - Valor

  El responsable de este artefacto es el **Product Owner**, que es el encargado de añadir y ordenar los ítems correspondientes.

- ***Sprint Backlog*:** Es un **subconjunto de elementos** del *Product Backlog* seleccionados para realizarse en un *sprint*. Este subconjunto es ofrecido como **incremento** del producto y lograr así el **objetivo del *Sprint***.

  Su función es visibilizar el tiempo necesario para alcanzar el objetivo establecido de manera que el Equipo de Desarrollo tenga conocimiento de éste.

- ***Sprint Burndown Charts*:** Es una representación gráfica que nos permite observar el avance de un sprint de manera que podemos ver el trabajo pendiente a lo largo del tiempo y ver la velocidad a la que se está completando los objetivos y requisitos.

  Es un artefacto muy útil para poder predecir cuando se finalizará el proyecto.

4.3 Aplicación de Scrum al proyecto

A pesar de lo comentado en el apartado 4.1, es prácticamente imposible cumplir todos los aspectos de la metodología *Scrum* en un Trabajo de Fin de Grado. Por ello, en esta sección hablaremos de las adaptaciones que se han realizado al proyecto en base a *Scrum*.

La primera incongruencia que encontramos con respecto al modelo *Scrum* es la realización y entrega de un anteproyecto que definía una estructura del proyecto completo más o menos fija, por lo que contrapone la flexibilidad de añadir nuevas metas y funcionalidades dependiendo de las necesidades del cliente. Siguiendo con esto, no existe un cliente como tal y no podemos obtener un *feedback* de éste, en este caso el cliente correspondería al tribunal de defensa que es el encargado de comentarnos tanto lo que está bien como lo que está mal en el, y ese feedback no se obtendría hasta la finalización del proyecto por lo que no se pueden ir arreglando ni cumpliendo las funcionalidades desde la visión del cliente sino que se tendría que hacer desde un juicio propio con ayuda de los directores.

En cuanto a los **roles**, el *Product Owner* se vería correspondido con los directos del Tabajo de Fin de Grado, que serían los encargados dirigir y de proporcionar las herramientas necesarias, en este caso componentes *hardware*, al Equipo de Desarrollo, que en esta adaptación sería compuesto por un único integrante que es el alumno. En el caso del *Scrum Master*, no tenemos ninguna figura representativa al alcance por lo que también será representada por los directores del Trabajo de Fin de Grado que se encargarán de abogar por el cumplimiento de la metodología y de eliminar los impedimentos que pudieran surgir.

En el caso de los **eventos** se han intentado cumplir en la medida de lo posible. Los ***Sprints*** se incluyeron como pequeños periodos de tiempo, normalmente de dos semanas. Antes de cada *Sprint* se realizaba una **Reunión de Planificación** de éste en el que se definían los objetivos y tareas que habría que llevar a cabo antes de la siguiente reunión. Las reuniones de ***Sprint Review*** también se realizaron, analizando los objetivos que se habían cumplido y lo que quedaba pendiente.

En el caso de las ***Daily Scrum*** fue un evento imposible de cumplir tanto por la disponibilidad de los directores como la del alumno.

Otro de los aspectos que se han definido de la metodología *Scrum* ha sido el ***Product Backlog*** en forma de anteproyecto, en este se definían los objetivos, herramientas y plan de trabajo para el proyecto y para su correcto desarrollo.

4.4 Estimación de recursos y costes

En esta sección haremos una lista de los recursos empleados en el proyecto, tanto humanos como materiales y se hará una estimación del coste total que tendría la realización de éste en

un entorno profesional.

4.4.1 Recursos

Ya se han comentado previamente lo recursos humanos utilizados en el apartado de roles de la metodología *Scrum* 4.2.1, éstos son:

- Product Owner: Directores del Trabajo de Fin de Grado
- Scrum Master: Directores del Trabajo de Fin de Grado
- Equipo de Desarrollo: Alumno

En el caso de los recursos materiales, han sido proporcionados tanto por parte del alumno como por parte de los profesores. Los más significativos son:

- Ordenador personal: *Portátil MSI PS63 Modern 8RC-013ES Intel Core i7-8565U*. Utilizado para el completo proceso de desarrollo y pruebas, valorado en **1.200€**.
- Monitor: *Monitor AOC 24B2XH* de 24" utilizado para una mayor comodidad por parte del desarrollador, valorado en **100€**.
- Adaptador de red externo: *Adaptador CSL-AC1200 USB 3.0 Dual Band WLAN Stick* utilizado para la visualización de tráfico de red, proporcionado por los directores y valorado en **30€**.
- Raspberry Pi 3 Model B+: Placa base utilizada para el despliegue y pruebas de la herramienta, proporcionada por los directores y valorada en **35€**.

4.4.2 Costes

Para calcular el coste de los recursos humanos se ha utilizado la Guía Salarial del Sector TI en Galicia 2015/16 [8]. Se ha supuesto que el desarrollador es un desarrollador Junior. En la figura 4.2 podemos ver el desglose del coste.

Recursos humanos	N.º miembros	Coste base (€/hora)	Tiempo Total (horas)	Total (€)
Desarrollador Junior	1	8,26	250	2 065
Director proyecto	2	20,84	50	2 084
			Subtotal	**4.149,00 €**

Figura 4.2: Desglose del coste de los recursos humanos

En el caso del coste de los recursos materiales se tendrán en cuenta todos los mencionados en el apartado 4.4.1. En la figura 4.3 podemos ver el desglose de estos costes.

Recursos materiales	Cantidad	Coste (€)	Total(€)
Ordenador personal	1	1 200	1 200
Adaptador de red	1	30	30
Monitor	1	100	100
Raspberry Pi	1	35	35
		Subtotal	**1.365,00 €**

Figura 4.3: Desglose del coste de los recursos materiales

Sumando el coste de los recursos humanos y recursos materiales nos saldría un **coste total de 5.514,00€**.

4.5 Planificación de Sprints

En el Product Backlog, que en este caso se corresponde con el anteproyecto, quedaron definidos cada uno de los *sprints* que se iban a realizar en el completo desarrollo del proyecto. A pesar de esto, debido a la naturaleza del Trabajo de Fin de Grado se ha tenido que modificar ligeramente el contenido y el objetivo de algunos sprints.

4.5.1 Diseño de la arquitectura software

Se realiza un diseño de la arquitectura *software* para establecer la estructura básica del proyecto. La finalidad de este *sprint* es la de conocer todos los elementos necesarios para el desarrollo del mismo, así como el orden y la manera en la que éstos tienen que ir colocados. Este *sprint* sirve de guía para el resto de *sprints* del proyecto.

4.5.2 Implementación del módulo de captura de información física

Se corresponde con el primer objetivo definido que es el de la detección y enumeración de redes inalámbricas a través del uso de un adaptador de red externo 3.6.1 y de la librería Scapy 3.1.2. Este es el paso clave para la implementación del resto de funcionalidades de la herramienta.

4.5.3 Extracción de información de las redes inalámbricas

Una vez tenemos ya implementada la funcionalidad de detección y enumeración de redes, el siguiente paso es extraer la información relevante de cada una de éstas. Para ello es importante poder clasificar la información en diferentes grupos: información básica, información de la señal e información de la seguridad.

4.5.4 Desarrollo del módulo de generación de inventario

En este *sprint* se busca desarrollar un módulo para la generación de un **inventario de dispositivos conectados** a una red a la que estaremos previamente autenticados. También implementado con Scapy 3.1.2

4.5.5 Programación de la interfaz Web

Se hace una primera aproximación del desarrollo de la interfaz Web teniendo ya las funcionalidades básicas implementadas. Se hace uso del *framework* Django 3.1.3 para su desarrollo.

Debido a la naturaleza de la herramienta, la interfaz web irá creciendo y adaptándose conforme se desarrollen nuevas funcionalidades.

4.5.6 Implementación del módulo de monitorización intensiva

En este *sprint* se desarrolla una nueva funcionalidad que no se había contemplado previamente. Se busca implementar un módulo de monitorización intensiva de manera que se pueda poner todo el **foco de atención** en una red inalámbrica para poder extraer información de una manera más detallada.

4.5.7 Desarrollo de la interfaz de comunicación con el servidor central

Se trabaja en un sistema de comunicación securizado entre el sistema de captura y el servidor centralizado para el **envío de información** mediante una API REST.

4.5.8 Despliegue en dispositivo físico

Hasta ahora todas las funcionalidades se habían probado en el ordenador personal del alumno y sin haber hecho uso de las herramientas software de despliegue 3.4.

Por lo tanto, en este *sprint* se hace un despliegue de la herramienta en la **Raspberry Pi** 3.6.2 haciendo uso de la herramienta **Docker** 3.4.1 para su instalación.

Capítulo 5

Desarrollo

A lo largo de este capítulo se detallará el proceso completo para el desarrollo del proyecto. Para empezar, se hablará sobre la arquitectura y estructura que sigue el proyecto y se continuará detallando cada uno de los módulos del sistema, analizando así su análisis, diseño e implementación.

5.1 Arquitectura del sistema

La arquitectura de la herramienta es simple, consiste en un **sistema distribuido** en el que existen uno o más dispositivos cuya función es monitorizar el tráfico de red, extrayendo así información relevante para posteriormente enviarla a un servidor centralizado.

La ventaja que nos proporciona esta arquitectura es que permite tener múltiples dispositivos de captura en diferentes áreas de un mismo entorno de manera que podremos obtener una gran cantidad de información de manera mucho más detallada y precisa.

Por lo tanto, como acabamos de comentar, el pilar fundamental del sistema son los dispositivos de captura. Cada dispositivo se compone de cuatro elementos: un sistema de escucha, un procesador de información, una base de datos y una aplicación web.

El **sistema de escucha** permite al dispositivo capturar todo el tráfico de red del área en el que se encuentra de forma pasiva mediante un adaptador de red externo 3.6.1. Muchas de las tramas que se encuentran en ese tráfico no son de gran relevancia para nuestra herramienta por lo que el dispositivo dispone de una funcionalidad de **procesado** que se encarga de procesar las tramas de red que nos son de utilidad y de extraer la información más importante de cada una de ellas.

Cada dispositivo dispone de una base de datos personal y debido a que el sistema no procesa todas las tramas ni guarda toda la información de ellas, sino que sólo procesa las tramas relevantes para extraer la información necesaria, el espacio que ocupa esa información en la base de datos no es muy grande.

Este dispositivo de captura también se encarga de contener una **aplicación web**, que será la encargada de mostrar toda la información al usuario mediante una interfaz web «amigable» y de realizar el envío de datos mediante peticiones web seguras al servidor central con los datos de cada dispositivo.

En resumen, cada dispositivo de captura tiene como función procesar y transformar toda la información recibida, guardarla en una base de datos y mostrarla mediante una interfaz web. Además, cada dispositivo se encargará de enviar la información almacenada cada cierto tiempo a un servidor central de manera que podremos tener toda la información de diferentes áreas de un entorno en un sistema centralizado.

Gracias a este sistema centralizado podemos tener una visión global de todo el entorno de manera precisa.

[esquema]

5.2 Módulo de captura de tráfico de red

Como habíamos comentado previamente, cada uno de los dispositivos de captura contiene un sistema de captura de tráfico de red mediante un adaptador de red externo.

El adaptador de red externo nos da las capacidades físicas para poder esnifar ese tráfico pero para poder manejarlo dentro de nuestra herramienta hemos utilizado la librería Scapy [1], cuya funcionalidad ya se ha explicado en el apartado 3.1.2.

En el listing 5.1 se muestra la función de captura de tráfico principal que, como podemos ver, llama a la función **sniff()** pertenenciente a la librería Scapy.

Esta función permite capturar de forma pasiva todo el tráfico de red que está a nuestro alcance. En este caso se han utilizado algunos parámetros para su funcionamiento: el primero de ellos *iface* determina la interfaz de red con la que se quiere capturar el tráfico, el parámetro *prn* es uno de los más importantes ya que permite pasar una función a cada uno de los paquetes capturados para poder manejarlos y extraer información de ellos, y por último *count*, que indica el número de paquetes que se quiere capturar como máximo, en este caso se pone un límite de 25 ya que nos interesa mostrar la información capturada cada cierto tiempo y si no se pone un límite, la información no se devuelve hasta que haya una interrupción por parte del usuario.

[1] https://scapy.net/

```
def start_sniffing(interface):

  p = Process(target=channel_hopper, args=(interface,))
  p.start()

  sniff(iface=interface, prn=sniffer, count=25)

  p.terminate()
  p.join()
```

Listing 5.1: Función de captura de tráfico

Como podemos ver, antes de llamar a la función **sniff**, se crea un objeto **Process** que sirve para crear un proceso multithread de una función pasada como *target*. En este caso se crean múltiples instancias de la función **channel_hopper()** que nos permitirá capturar el tráfico de red de forma concurrente de todos los canales disponibles.

```
def channel_hopper(interface):
  list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 36, 40,
    44, 48]
  while True:
    try:
      channel = random.choice(list)
      os.system("iwconfig %s channel %d" % (interface, channel))
      time.sleep(1)
    except:
      break
```

Listing 5.2: Función *channel_hopper*

La función **channel_hopper()** 5.2 consiste en un bucle infinito en el que se va escogiendo de forma aleatoria un número de una lista previamente definida que coincidirá con el canal en el que se colocará la antena Wi-Fi para capturar el tráfico de red.

Básicamente se crea una lista con los canales soportados por la antena Wi-Fi y mediante un bucle infinito y una llamada al sistema se cambia cada segundo el canal en el que se encuentra la interfaz de red capturando tráfico. Si no se especifica este cambio de canal, la interfaz de red se quedaría capturando tráfico estáticamente en un sólo canal y perderíamos una gran cantidad de información.

5.3 Análisis del tráfico de red

Una vez tenemos un módulo de captura de tráfico de red, el siguiente paso es analizar ese tráfico de red capturado clasificándolo según las necesidades del proyecto.

5.3.1 Detección de redes y extracción de información

El primer objetivo de la herramienta que sirve como pilar fundamental para el resto de funcionalidades es claro, necesitamos un módulo que nos permita detectar las redes inalámbricas disponibles y enumerar sus características. Para ello necesitamos encontrar un tipo de tramas de red que nos permita extraer la información de cada uno de los AP de las redes Wi-Fi.

Como hemos comentado en el apartado 2.2 existen un tipo de tramas que son las llamadas *Beacon Frames* que se encargan de anunciar la presencia de la red WLAN periódicamente. Estas tramas se envían en claro y contienen toda la información de la red inalámbrica por lo que son las elegidas para descubrir y extraer los datos de cada punto de acceso de la red.

Como hemos dicho, la función **sniff** comentada en el apartado 5.2 sirve para escuchar todo el tráfico de red que está transitando en tiempo real, pero el parámetro *prn* es el que nos permite manipular ese tráfico capturado mediante una función suplementaria .

La función que hemos definido se llama **sniffer()** 5.3 y recibe cada uno de los *packets* capturados mediante el argumento *p* para su posterior manipulación. El flujo es el siguiente:

```
1 def sniffer(p):
2
3     time = str(datetime.today().strftime("%Y-%m-%d %H:%M:%S"))
4
5     if p.haslayer(Dot11):
6         if p.haslayer(Dot11Beacon):
7             beacon_packet(time, p)
```

Listing 5.3: Funcionalidad de capturar *Beacon Frames*

1. La función **sniffer** recibe un *packet* como argumento.
2. Mediante el método **datetime.today()** se guarda un *timestamp* de cada uno de los datos recibidos para tener constancia.
3. Se comprueba que la trama recibida pertenezca al estándar IEEE 802.11 (WLAN) mediante el método **haslayer(Dot11)** y posteriormente se comprueba que esa trama Wi-Fi sea una trama de tipo *Beacon* mediante el método **haslayer(Dot11Beacon)**, ambos métodos pertenencientes a la clase **scapy.packet**.
4. Se llama a la función **beacon_packet()** que será la encargada de procesar las tramas de tipo *Beacon* y de extraer los datos pertinentes.

La función **beacon_packet()** se encarga de extraer cada una de las características de la red Wi-Fi mediante el análisis de la trama de tipo *Beacon*.

Son varias las características que necesitamos extraer para hacer un reconocimiento a gran escala de cada una de las redes inalámbricas del entorno.

Scapy [2] nos proporciona una gran cantidad de herramientas y métodos para la extracción de estas características. A continuación se expondrán cada una de las características necesarias y el método de obtención de cada una de ellas

Características generales

- **SSID:** Corresponde al nombre de la red inalámbrica y en éste caso es una de las características más identificativas de cada red a pesar de que puedan existir varias con el mismo nombre.

```
raw_ssid = packet[Dot11Elt].info.decode('utf-8')
ssid = raw_ssid.strip() if '\x00' not in raw_ssid and raw_ssid !=
    '' else '[HIDDEN SSID]'
```

Listing 5.4: Obtención SSID

  La variable **raw_ssid** accede al campo del paquete donde se encuentra el SSID y lo formatea de manera que sea legible para el usuario. Puede ser que el SSID esté oculto y en ese caso la información obtenida es una cadena de múltiples caracteres en hexadecimal, por lo que la variable **ssid** se encarga de comprobar si es el SSID se muestra en claro o no, y en este último caso la bautiza como *HIDDEN SSID*.

- **BSSID:** Corresponde con el identificador de cada red inalámbrica, en este caso si debería ser único ya que se forma con la dirección MAC del dispositivo de punto de acceso. De la misma manera que el anterior, simplemente se accede al campo de la trama que corresponde con el BSSID que en este caso es el campo **addr3**.

```
bssid = packet[Dot11].addr3
```

Listing 5.5: Obtención BSSID

- **Manufacturer:** El *Manufacturer* indicar el vendedor oficial del dispositivo que funciona como punto de acceso de la red inalámbrica. Esta característica se puede obtener a partir de las tres primeras tuplas de la dirección MAC del dispositivo.

```
manufacturer = utils.manf(bssid.upper()[:8])
```

Listing 5.6: Obtención Manufacturer

[2] https://scapy.net/

```
def manf(mac):
  with open('mac-vendors-export.json') as f:
    vendor = json.load(f)

    for p in vendor:
      if p['macPrefix'] == mac:
        return p['vendorName']

  return "Unknown"
```

Listing 5.7: Función de apoyo manf()

manf() es una función que recibe los ocho primeros caracteres de la dirección MAC del dispositivo (que corresponde con las tres primeras tuplas) como argumento y devuelve el vendedor correspondiente a partir de un archivo en JSON que contiene las correspondencias del vendedor con cada dirección MAC. En caso de no encontrar la correspondencia se devuelve *Unknown*.

- **Supported Rates:** Los *Supported Rates* se corresponde con la velocidad de datos soportada por cada punto de acceso. Para obtener estos *rates* se utiliza un método personalizado creado por Scapy llamado **network_stats()** que agrupa las estádisticas básicas de cada punto de acceso en forma de diccionario.

```
nstats = packet[Dot11Beacon].network_stats()
rates = nstats['rates']
```

Listing 5.8: Obtención Supported Rates

- **Beacons:** Es el número de tramas de tipo *beacon* que ha generado un punto de acceso desde el momento en el que se comienza a capturar el tráfico de la red. Para esto simplemente se crea un contador que lleva la cuenta del número de *Beacon frames* capturados.

Características de señal

Se corresponden con las características de la señal que emite el AP. Para ello se hace uso de la cabecera **Radiotap** que agrupa toda la información relacionada con la señal emitida. Para hacer uso de esta cabecera simplemente se obtiene la capa **Radiotap** de cada una de las tramas con el método **getlayer()**.

```
radiotap = packet.getlayer(RadioTap)
```

Listing 5.9: Obtención de la capa Radiotap

- **RSSI:** El RSSI es el indicador de potencia de señal emitida más común en redes Wi-Fi y se mide en dBm. Para obtenerlo simplemente hay que hacer uso de la capa **Radiotap** obtenida anteriormente.

```
rssi = radiotap.dBm_AntSignal
```

Listing 5.10: Obtención del RSSI

- **Canal:** Indica el canal en el que se encuentra la red Wi-Fi emitiendo su señal. Como hemos comentado en el apartado 2.4, lo más común es que una red inalámbrica no se encuentre de forma estática en un único canal sino que puede estar saltando de canal en canal buscando unas mejores condiciones de señal. A pesar de que el canal sea cambiante en la mayoría de los casos, cada red Wi-Fi suele disponer de un canal central que toma referencia para esos saltos de canales.

- **Canal Central:** Para obtener este canal central, el AP toma como referencia la frecuencia de señal en la que se encuentra emitiendo la red Wi-Fi.

- **Frecuencia:** Cada red Wi-Fi puede estar emitiendo en una frecuencia distinta. Estas frecuencias están previamente definidas y pueden depender de cada país.

```
frecuency = radiotap.ChannelFrequency

central_channel = channel_frecuency[frecuency]

try:
  channel = packet[Dot11EltDSSSet].channel
except:
  channel = central_channel
```

Listing 5.11: Obtención Frecuencia, Canal central y Canal

  La frecuencia se obtiene directamente de la capa **Radiotap** accediendo al campo correspondiente. Para obtener el canal central se hace uso de un diccionario de correspondencias frecuencia-canal y se le pasa la frecuencia obtenida como argumento y por último, el canal actual se obtiene accediendo a la capa **Dot11EltDSSSet**, y en caso de que el parámetro DSSS no esté activado, el canal será igual al canal central.

- **Ancho de banda:** Las redes Wi-Fi no emiten su señal en una única frecuencia sino que para evitar el solapamiento entre redes, éstas disponen de un ancho de banda que les permite emitir en una frecuencia u otra dependiendo de la saturación de la red. Normalmente, las redes 2.4Ghz tienen un ancho de banda de 20Mhz y las redes 5Ghz de 40Mhz. Haciendo uso de la capa **Dot11EltHTCapabilities** de la trama se accede al

campo Supported_Channel_Width que nos devuelve 0 o 1 dependiendo del ancho de banda soportado y haciendo uso de un diccionario obtenemos el valor correspondiente.

```
cb_layer = packet[Dot11EltHTCapabilities].Supported_Channel_Width

channel_bandwidth = channel_width[cb_layer][:-3]
```

Listing 5.12: Obtención del Ancho de banda

*Se eliminan los últimos tres caracteres que corresponden a la unidad de medida.

- ***Spectrum*:** Se corresponde a la **banda de frecuencia** en la que se encuentra la red Wi-Fi. Como hemos adelantado previamente existen dos bandas de frecuencia principales: 2.4Ghz y 5Ghz. Éstas últimas son las más actuales y disponen de una mejora respecto a la banda de frecuencia 2.4Ghz.

```
channelflags = str(radiotap.ChannelFlags)

spectrum = channelflags.split('+')[1]

if spectrum[0] == '2':
    spectrum = '2.4Ghz'
```

Listing 5.13: Obtención del *Spectrum*

Haciendo uso de la capa **Radiotap** se accede a una serie de datos llamados ChannelFlags que corresponden con ciertos *flags* relacionados con el canal de la red Wi-Fi. En este caso, Scapy reconoce la banda de frecuencia 2.4Ghz como 2Ghz por lo que se hacen los arreglos pertinentes.

- **FSPL:** Free-Space Path Loss (FSPL) es la atenuación de la señal de radio entre dos antenas que resulta de la combinación del área de captura de la antena receptora más la trayectoria de línea de visión libre de obstáculos a través del espacio libre. En pocas palabras, es una aproximación de la distancia entre el AP de la red Wi-Fi y nuestro dispositivo de captura. Es posible calcular esta distancia a partir del RSSI y la frecuencia de la señal obtenida [9].

$FSPL = 20\log_{10}(d) + 20\log_{10}(f) - 27.55$

Siendo d la distancia y f la frecuencia. El resultado está expresado en metros.

```
distance = (27.55 - (20 * math.log10(frecuency)) + math.fabs(rssi))
    / 20.0
fspl = math.pow(10, distance)
```

Listing 5.14: Obtención del FSPL aplicado al RSSI

Características de Seguridad

Gracias a estas características podemos obtener información de la configuración de seguridad que tiene cada red Wi-Fi.

- **Protocolo de seguridad:** Como hemos comentado en el apartado 2.4, son varios los protocolos de seguridad que existen hasta la fecha y estos son el primer indicador del nivel de seguridad de una red inalámbrica.

```
crypto = []
for element in nstats['crypto']:
    var = element.split("/")[0]
    crypto.append(var)
```

Listing 5.15: Obtención del protocolo de seguridad

  Hacemos uso de la variable **nstats** obtenida en el listing 5.8 para obtener los protocolos de seguridad. Cada red Wi-Fi puede contener más de un protocolo de seguridad, es por eso que se va añadiendo a una nueva lista cada uno de los protocolos que pueda contener.

- **Sistema de autenticación:** Pueden ser varios los sistemas de autenticación que utiliza una red Wi-Fi para autenticar a sus dispositivos. El más común para redes domésticas es Pre-Shared Key (PSK) que consiste en una clave secreta compartida con anterioridad entre las dos partes a través de un canal seguro.

```
suite = suite[packet[AKMSuite].suite]
```

Listing 5.16: Obtención del sistema de autenticación

- **Algoritmo de cifrado:** Es el sistema que utiliza la red inalámbrica para cifrar la información emitida por nuestros dispositivos.

```
cipher = cifrado[packet[RSNCipherSuite].cipher]
```

Listing 5.17: Obtención del algoritmo de cifrado

  Tanto el **sistema de autenticación** como el **algoritmo de cifrado** se obtienen haciendo uso de dos capas ofrecidas por Scapy que devuelven un resultado en hexadecimal y a través de un diccionario se devuelve el resultado en claro.

5.3.2 Detección de dispositivos conectados a la red sin autenticación previa

Uno de los aspectos más importantes a la hora de enumerar redes inalámbricas es poder obtener información sobre los dispositivos que estén conectados a éstas. Esto es posible gracias al análisis de tráfico generado por la comunicación entre un AP y un dispositivo.

Ya hemos adelantado en el apartado 2.2 los cuatro tipos de tramas que existen en esta capa de red y hemos explicado la importancia de las tramas de **Gestión** para la extracción de información de las redes inalámbricas.

Las tramas de **Datos** son las encargadas de transportar la información de una estación a otra dentro de una misma red, y las tramas de **Control** trabajan en conjunción con las anteriores para transportar los datos de manera confiable.

Con la descripción anterior sabemos que si podemos obtener información de los dispositivos de una red sin estar previamente autenticados va a ser gracias a las tramas de Datos y de Control.

Como habíamos mostrado en el apartado 5.3.1 en el listing 5.3, tenemos una función llamada **sniffer()** que se encarga de clasificar el tráfico entrante para su posterior manipulación. Como podemos ver en en el listing 5.18 se ha añadido una nueva clasificación a la función que se encarga de capturar las tramas de tipo 1 y 2 correspondientes con las tramas de Control y de Datos respectivamente.

```
1 def sniffer(p):
2
3     time = str(datetime.today().strftime("%Y-%m-%d %H:%M:%S"))
4
5     if p.haslayer(Dot11):
6         if p.haslayer(Dot11Beacon):
7             beacon_packet(time, p)
8         elif p.getlayer(Dot11).type in [1, 2]:
9             datacontrol_packet(p)
```

Listing 5.18: Clasificación de tramas añadida

La función **datacontrol_packet()** se encarga de procesar cada una de las tramas capturadas que sean de de Datos o de Control.

```
1 def datacontrol_packet(packet):
2
3   ra = packet.getlayer(Dot11).addr1
4   sa = packet.getlayer(Dot11).addr2
```

Listing 5.19: Función datacontrol_packet()

La única información que podemos obtener con este tipo de tramas es la dirección MAC de cada dispositivo, por eso, en este caso se trata los dos tipos de tramas de la misma manera

ya que ambas poseen una Receiver Address (RA) y una Sender Address (SA).

En cambio, existen una serie de problemas a la hora de identificar estas direcciones MAC:

- Una de las dos direcciones MAC comentadas previamente (RA y SA) va a ser la del punto de acceso de la red Wi-Fi, por lo tanto hay que **distinguir cual es la del punto de acceso y cual es la del dispositivo**. Como la comunicación entre AP y dispositivo es recíproca, no sabemos si la información es enviada por el AP y recibida por el dispositivo o viceversa.

  Para solucionar esto, se crea una lista de puntos de acceso analizados previamente y se comprueba si una de las dos direcciones MAC se encuentra en esa lista y por concluyente, la otra dirección MAC va a ser la del dispositivo.

- La comunicación entre AP y dispositivo no es el único tráfico de éste tipo que transita por la red. También se envían tramas que tiene como dirección MAC de destino una dirección Broadcast o Multicast o incluso direcciones MAC de protocolos como Spanning Tree Protocol (STP).

  Esto solo añadiría ruido a nuestro sistema y crearía dispositivos que no existen. Para ello se crea una función llamada **noise_filter()** que toma las dos direcciones MAC como argumento y detecta si son direcciones de ”ruido”.

```
1    if not utils.noise_filter(sa, ra):
2        if sa in ap_list:
3            if ra not in cliente_usado:
4                client_dict = {'mac': ra, 'connected_to': sa,
    'manufacturer': utils.manf(ra.upper()[:8])}
5                ap_dict[sa]['clients'].append(client_dict)
6                cliente_usado.append(ra)
7
8        elif ra in ap_list:
9            if sa not in cliente_usado:
10               client_dict = {'mac': sa, 'connected_to': ra,
    'manufacturer': utils.manf(sa.upper()[:8])}
11               ap_dict[ra]['clients'].append(client_dict)
12               cliente_usado.append(sa)
```

Listing 5.20: Filtración de direcciones MAC de dispositivos

En el listing 5.20 se muestran todos los pasos para detectar cual es la dirección MAC del dispositivo y a que red inalámbrica está conectado. También, mediante la función **manf()** que habíamos comentado previamente, se detecta el vendedor del dispositivo.

Apéndices

Lista de acrónimos

AJAX Asynchronous JavaScript And XML. 17

AP Wireless Access Point. 7–12

API Application Programming Interfaces. 2, 29

BSSID Basic Service Set Identifier. 12

CSS Cascading Style Sheets. 16

dBm Decibelio-milivatio. 12

DOM Document Object Model. 17

DoS Denial of Service. 9, 12

HTML HyperText Markup Language. 16

IEEE Institute of Electrical and Electronic Engineers. 7, 8

MAC Media Access Control. 9, 10, 12

MitM Man-in-the-Middle. 12

MVC Model View Controller. 16

MVT Model View Template. 16

PyPI The Python Package Index. 16

REST Representational State Transfer. 2, 29

RSSI Received Signal Strength Indicator. 12

SSID Service Set Identifier. 12

TI Tecnologías de la Información. 1–3

USB Universal Serial Bus. 7

WLAN Wireless Local Area Network. 2, 4, 8

WNIC Wireless Network Interface Controller. 7

WPA Wi-Fi Protected Access. 10, 11

WSGI Web Server Gateway Interface. 19

YAML YAML Ain't Markup Language. 18

Glosario

base de datos . 18, 20

broadcast . 9

channel hopping . 13

evil twin . 12

fuerza bruta . 10

handshake . 10, 11

modelos Definen la estructura de los datos almacenados en Django. 18

modo monitor . 2, 7

portal cautivo . 12

spoofear . 9

Wi-Fi . 17

Bibliografía

[1] D. of Computer Science and E. J. I. of Technology, "Analysis of various packet sniffing tools for network monitoring and analysis." [En línea]. Disponible en: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.429.567&rep=rep1&type=pdf

[2] A. A.-O.-b. Tariq AL-Kadia, Ziyad AL-Tuwaijrib, "Arduino wi-fi network analyzer." [En línea]. Disponible en: https://www.sciencedirect.com/science/article/pii/S1877050913008661

[3] M. S. Gast, "802.11 wireless networks: The definitive guide." [En línea]. Disponible en: https://vdocuments.site/guia-salarial-sector-ti-galicia-2015-2016.html

[4] P. Barry, "Head first python." [En línea]. Disponible en: https://oiipdf.com/head-first-python-2nd-edition

[5] "Scapy." [En línea]. Disponible en: https://scapy.readthedocs.io/en/latest/

[6] "Vis.js." [En línea]. Disponible en: https://visjs.github.io/vis-network/docs/network/

[7] J. Sutherland and K. Schwaber, "The scrum guide." [En línea]. Disponible en: https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

[8] "Guía salarial sector ti galicia." [En línea]. Disponible en: https://vdocuments.site/guia-salarial-sector-ti-galicia-2015-2016.html

[9] "Comparing ray tracing, free space path loss and logarithmic distance path loss models in success of indoor localization with rssi." [En línea]. Disponible en: https://ieeexplore.ieee.org/abstract/document/6143552