SISTEMA ALEJANDRA

Miguel Ángel Martínez de Castilla de la Mata
Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación
Área de Redes Inalámbricas

Director: Ferran Adelantado i Freixer

02 de junio de 2021

AGRADECIMIENTOS

"A mi madre, por sus largas jornadas de cuidados y amor, por haber sido siempre mi apoyo incondicional y enseñarme que nunca hay que rendirse ante las adversidades de la vida. Por siempre creer en mi. A mi padre, por su fuerza, sus cuidados benevolentes y llenos de cariño, su ejemplo de constancia, perseverancia y por nunca dejarme ganar a nada, por enseñarme que el esfuerzo con inteligencia es el mejor de los recursos. Por ser los mejores padres que se pueden tener. A mi mujer, por tanto amor, por su apoyo durante toda esta andadura y por estar siempre a mi lado. Porque con ella soy más fuerte y mejor persona. Por hacerme sentir el más afortunado del mundo. Y a mi hijo, por ser la luz que ilumina nuestra vida cada día con su sonrisa, por ser una fuente inagotable de alegría y amor que me hace querer ser siempre mejor."

FICHA DEL TRABAJO FINAL

Título del trabajo:	Sistema ALEJANDRA
Nombre del autor:	Miguel Ángel Martínez de Castilla de la Mata
Nombre del consultor/a:	Ferran Adelantado i Freixer
Nombre del PRA:	Ferran Adelantado i Freixer
Fecha de entrega (mm/aaaa):	06/2021
Titulación:	Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación
Área del Trabajo Final:	Área de Redes Inalámbricas
Idioma del trabajo:	Español
Palabras clave	“AI”, “SEGURIDAD”, “WIFI”

Resumen del Trabajo (máximo 250 palabras): Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.

La finalidad de este proyecto es el diseño de un sistema de seguridad que detectará posibles amenazas sobre una instalación sensible. Este sistema estará permanentemente escaneando el espectro WIFI en busca de dispositivos no autorizados.

Para ello, se hará un análisis de los paquetes escaneados en busca de sus peticiones de redes conocidas y su dirección MAC. Una vez obtenidos los datos, se pasará la MAC junto con el resto de las características por un algoritmo de machine learning que será capaz de discernir si se trata de un dispositivo no autorizado.

Con las redes conocidas de este dispositivo, se realizará un análisis superficial de fuentes abiertas en el que se obtendrá su ubicación y de esa manera determinar su potencial peligro.

Para la realización del proyecto se llevará a cabo primero una investigación sobre las tecnologías involucradas y las necesidades para su desarrollo.

Una vez completada, se desarrollará el código del programa mediante el lenguaje de programación Python que se implementará en un dispositivo que hará las veces de servidor.

Abstract:

The purpose of this project is the design of a security system that will detect possible threats on a sensitive installation. This system will be permanently scanning the WIFI spectrum in search of unauthorized devices.

To do this, the scanned packets will be analysed for their known network requests and MAC addresses. Once the data is obtained, the MAC will be passed along with the rest of the characteristics through a machine learning algorithm that will be able to discern whether it is an unauthorized device.

A superficial analysis in open sources will be done with the known networks in order to determine their location and potential danger.

To build the project, an investigation is firstly done over the involved technologies and the needs for their development.

Once completed, the program code will be developed using the Python programming language that will be implemented on a device that will act as a server.

Índice de contenido

Índice de figuras

Índice de tablas

1. Introducción

1.1 Contexto y justificación del trabajo

En la actualidad, debido al aumento del uso de las nuevas tecnologías en los conflictos bélicos, se crea una necesidad de protección que sea capaz de abarcar ese nuevo campo. En este sentido, el incremento de la utilización de la tecnología wifi para todo tipo de comunicaciones, tanto profesionales como personales, es un campo que se aborda en este proyecto como una medida más de protección de estructuras sensibles.

Este sistema de protección se encargará de identificar posibles amenazas sobre una instalación sensible mediante el estudio del espectro wifi mencionado anteriormente.

El diseño realizará un escáner permanente del espectro wifi en búsqueda de los paquetes emitidos por dispositivos móviles de las personas que se aproximen a las instalaciones. Estos dispositivos serán comparados con una *whitelist* que contenga los dispositivos permitidos y, en caso de detectar un intruso, se realizará un estudio del dispositivo por sus paquetes *ProbeRequest* que tienen información de sus redes conocidas. Una vez obtenidas se hará un pequeño análisis preliminar de su ubicación en la plataforma Wigle. Con ello se puede obtener una vista preliminar de los lugares comunes del dispositivo, y de esa manera valorar si pudiera suponer una amenaza potencial. Este resultado será comunicado mediante un email al responsable de seguridad.

En su versión SMART, la aplicación implementará un algoritmo de *machine learning* en la detección de las posibles amenazas, no solo para analizar los dispositivos que no se encuentren en la *whitelist*, si no con la finalidad de encontrar también posibles manipulaciones o suplantaciones de dispositivos existentes.

1.2 Objetivos del trabajo

- El estudio actual del estándar 802.11 así como la composición de los paquetes de comunicación wifi.
- Estudio superficial de la tecnología *machine learning*, con el fin de hallar algoritmos que se ajusten a nuestras necesidades.
- Desarrollo de la aplicación, así como la implementación del algoritmo *machine* learning, que sea capaz de detectar, analizar y comunicar vía correo electrónico la intrusión de una amenaza potencial.

1.3 Planificación del Trabajo

Para la realización del proyecto se seguirá el siguiente Diagrama de Gantt:

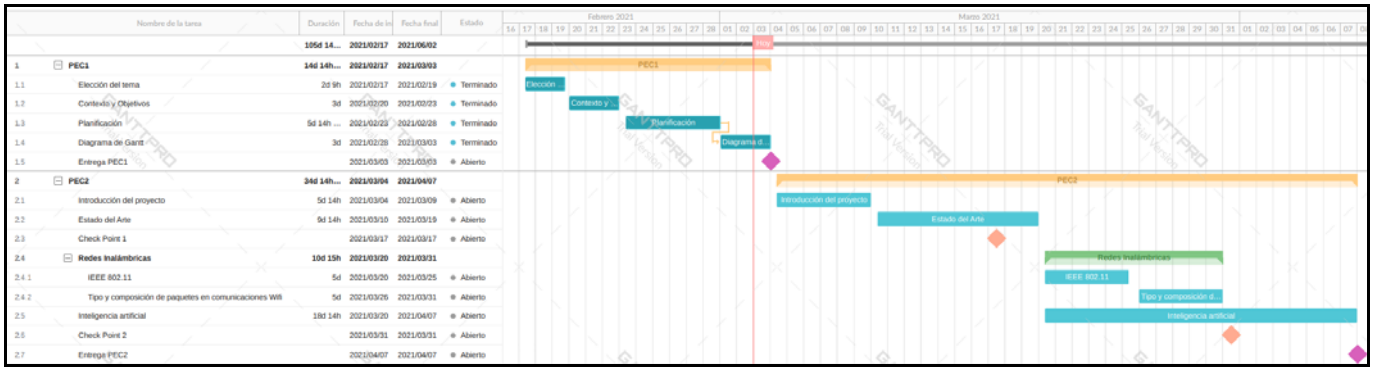


Figura 1 - Diagrama de Gantt (Parte I)

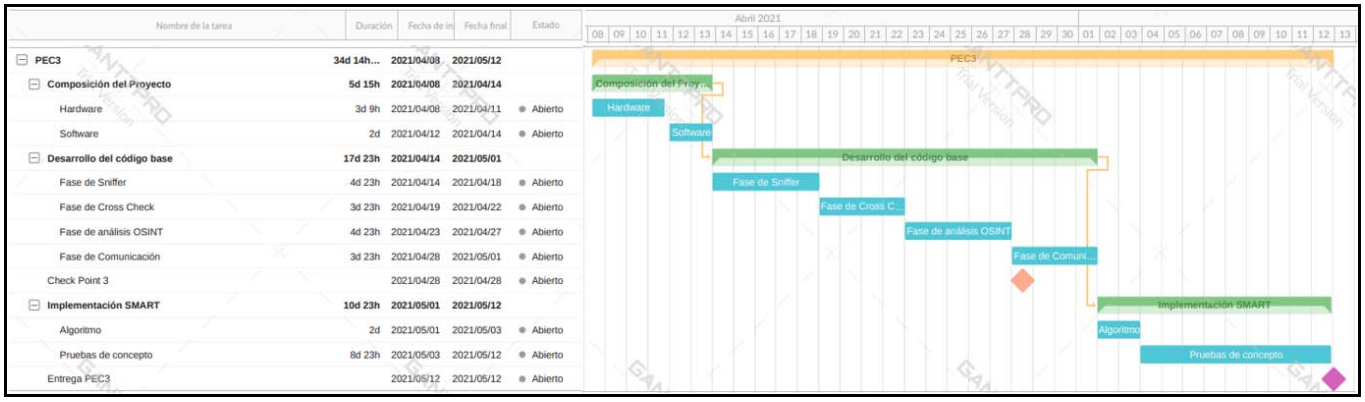


Figura 2 - Diagrama de Gantt (Parte II)

Figura 3 - Diagrama de Gantt (Parte III)

1.4 Breve sumario de productos obtenidos

- Dispositivo Raspberry PI 4B, que alojará el software haciendo las veces de servidor.
- Tarjeta de red ALFA AWUS1900.

1.5 Breve descripción de los otros capítulos de la memoria

El proyecto se divide en una introducción, dos capítulos que componen el cuerpo del proyecto y por último unas conclusiones.

El capítulo 1 incluye la parte más teórica del proyecto y está compuesto por tres puntos:

- **Estado del Arte:** Como su nombre indica se realiza acopio de información y se desarrolla el estado del arte del proyecto.
- **Redes inalámbricas:** En esta fase se realiza un trabajo de investigación, primero sobre el estado actual del estándar 802.11 ya que es la tecnología de comunicación que se va a estudiar en este proyecto, así como de la composición de los paquetes vía wifi, viendo que tipos hay, y cómo se diferencian, cómo están formados y cómo se puede aprovechar esta información para el objetivo del proyecto.
- **Inteligencia artificial:** En este punto, y debido a la amplitud de esta temática, se hace una breve investigación sobre cómo funciona la inteligencia artificial y se consultan fuentes para encontrar algún algoritmo ya definido que pueda servir como solución para la fase de detección del proyecto.

El capítulo 2 por otra parte incluye la fase práctica del proyecto, desde el hardware al desarrollo del programa, y está compuesto por los siguientes apartados:

- **Composición del proyecto**: En esta fase se eligen el *hardware* que compone el proyecto, así como una explicación de las tecnologías *software* utilizadas para el desarrollo del mismo (lenguajes de programación, librerías, aplicaciones externas, APIs, etc.)

- **Desarrollo de código base**: Aquí se realiza el grueso de desarrollo de código del proyecto. Estará formado por cuatro fases, la primera de obtención de datos; la siguiente en la que se crucen los datos obtenidos con una base de datos en la que estará almacenada una *whitelist*; la tercera, tras la detección será desarrollar el código encargado de realizar un estudio con fuentes abiertas de información sobre las ubicaciones de las redes conocidas y plasmar las mismas en un mapa; por último, se desarrollará la parte del código que realice la comunicación con el responsable de seguridad informando de la amenaza y los datos obtenidos del análisis.

- **Implementación SMART**: En esta fase se implementa el código necesario para realizar un algoritmo de *machine learning* para la fase de detección con el que se detecten falsos positivos. También se realizan pruebas de concepto mediante un *datasheet* semi artificial.

Por último, el apartado de conclusiones incorpora un resumen del proyecto con los datos más importantes y con posibles futuras vías de investigación y ampliaciones.

1.6 Estructura del sistema por fases de desarrollo

- Fase 1: Detección

El sistema monitoriza el entorno Wifi de las instalaciones mediante sensores, recogiendo los paquetes que emiten todos los dispositivos que se encuentran cerca.

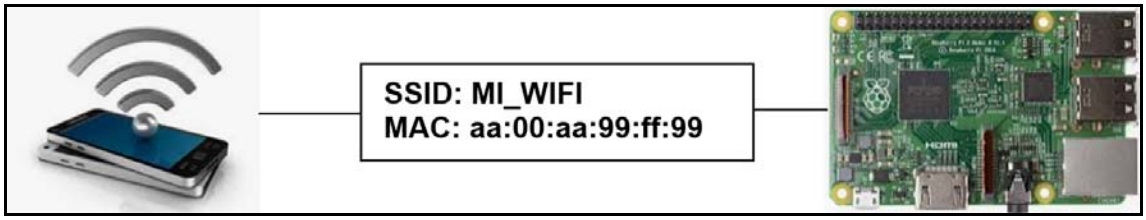


Figura 4 – Detección

- Fase 2: Procesado

Mediante software el sistema filtra los dispositivos de confianza estipulados en su Whitelist. Una vez localizado el posible intruso, se hace un estudio de sus peticiones en búsqueda de redes WIFI conocidas almacenándolas en una base de datos.

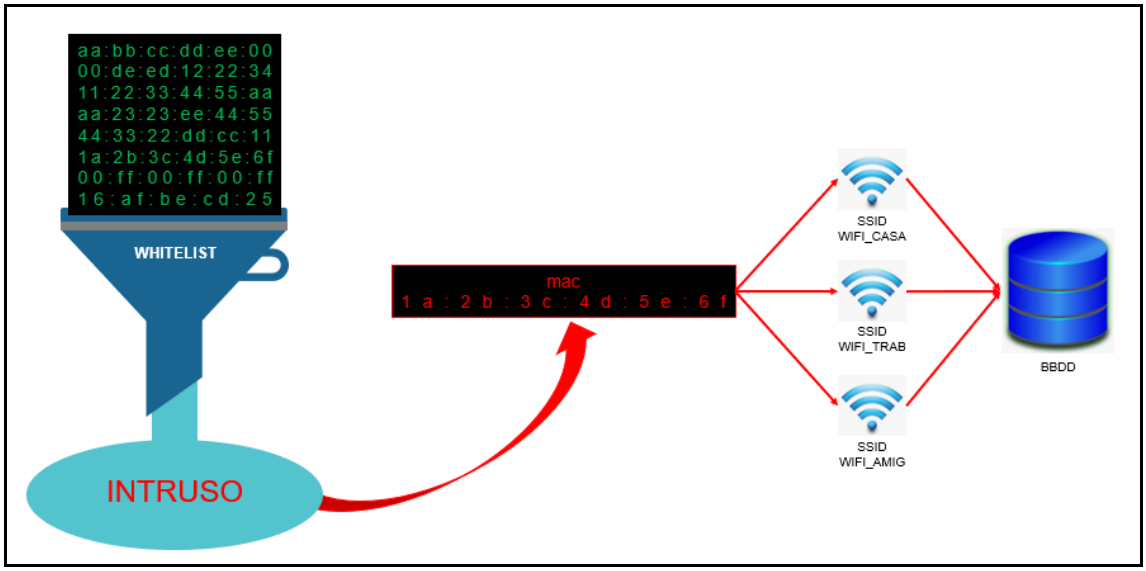


Figura 5 – Procesado

- Fase 3: Análisis OSINT

Una vez obtenidos los SSID del objetivo, se lanza una consulta a través de la API de la plataforma WIGLE[1] de la que obtendremos las coordenadas de las redes habituales del objetivo para continuar posteriormente con una investigación más profunda.

Figura 6 - Análisis OSINT

- Fase 4: Comunicación

Una vez obtenidos los datos del análisis el sistema se pondrá en contacto con el responsable de seguridad mediante correo electrónico, comunicando la alarma de proximidad y el resultado del análisis realizado.

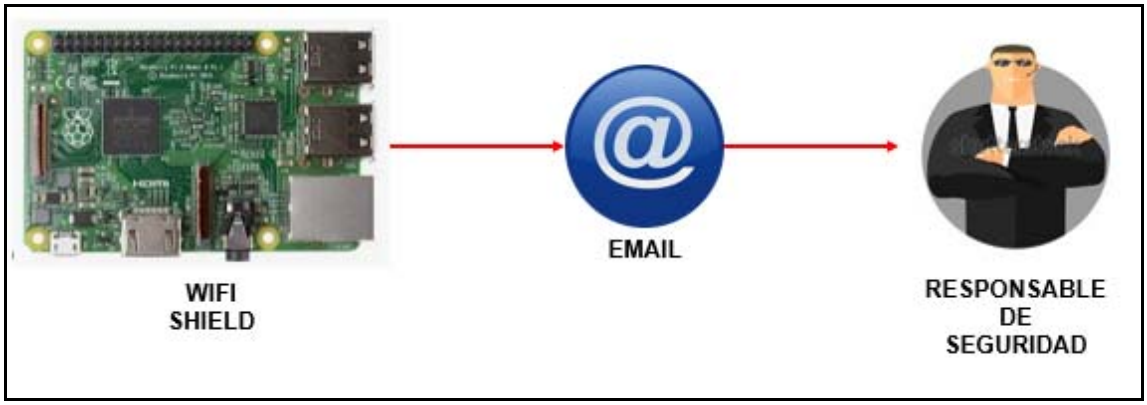


Figura 7 – Comunicación

[1] WIGLE. https://www.wigle.com

- Ejemplo de montaje del sistema

A continuación, se plasma un pequeño organigrama de funcionamiento del sistema:

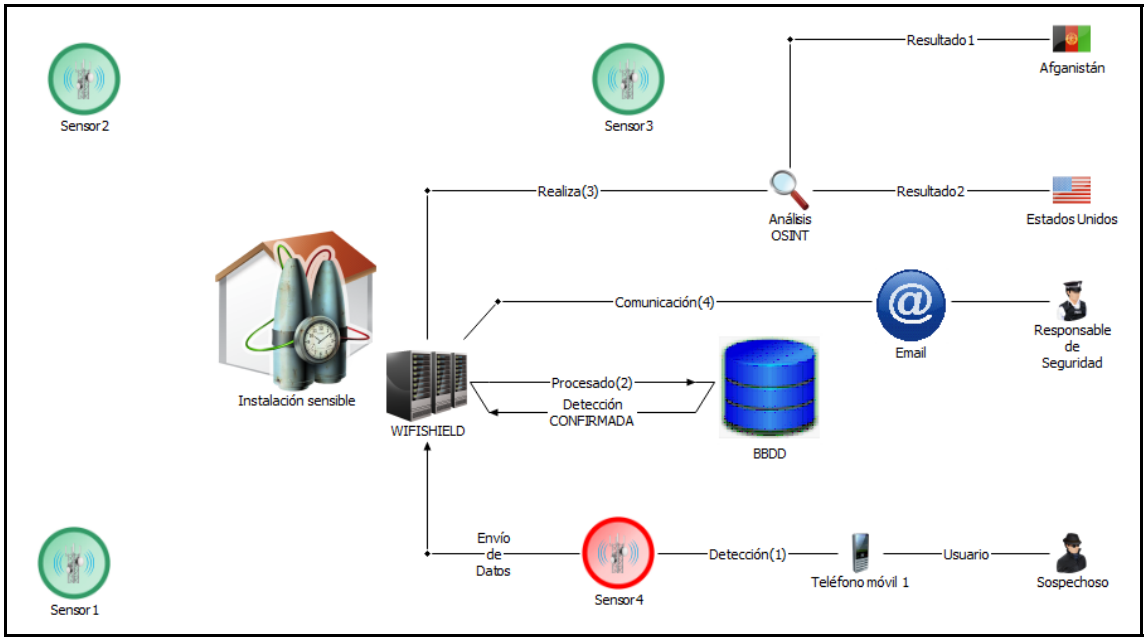


Figura 8 - Organigrama del sistema

2. Análisis Tecnológico

2.1 Estado del Arte

En todo conflicto bélico, el ataque o sabotaje a infraestructuras sensibles es altamente probable. Se consigue mermar la capacidad del enemigo tanto en comunicaciones como con desgaste de la ciudadanía. Es por ello, que el objetivo principal de este proyecto es proteger las necesidades de la ciudadanía, ante todo. Las principales instalaciones objetivo para este tipo de ataques son centrales eléctricas o instalaciones de comunicaciones.

Figura 9 - Instalaciones sensibles

Si bien en la actualidad existen numerosos sistemas de protección contra infraestructuras, este proyecto busca dar un paso más en esta seguridad introduciéndose en una capa invisible al ojo humano como es la de las ondas Wifi. Podemos asimilar este sistema a un escudo que escanea el entorno inalámbrico de una infraestructura.

Este sistema pretende aprovechar una debilidad de los posibles atacantes, que reside en su parte más humana, y esa es algo que en la actualidad está en pleno auge, el uso del teléfono móvil. El uso del dispositivo móvil ha sufrido un crecimiento exponencial en los últimos años, concretamente en España el 91.7%[2] de la población utiliza Smartphones para conectarse a internet, y, de hecho, el 70% del tiempo navegado se ha hecho desde algún dispositivo móvil. Quedando patente, que los dispositivos móviles están altamente ligados al día

[2] INE. https://www.ine.es/prensa/tich_2017.pdf

a día de cualquier ser humano y que es muy probable que los posibles atacantes lleven encima un teléfono móvil.

Para aprovechar esta debilidad, se estudia la tecnología wifi y como están formadas sus comunicaciones, poniendo en este caso especial interés sobre los paquetes de tipo *Probe Request*. Estos paquetes son enviados por los dispositivos para realizar pruebas de conexión sobre sus redes conocidas, y en caso de encontrarse cerca una de ellas, ésta le contesta con un *Probe Response* para comenzar el inicio de protocolo de conexión.

Hoy en día existen numerosos sistemas de seguridad que se encargan de proteger cualquier tipo de infraestructura. Desde las rudimentarias alarmas basadas en ultrasonidos con sensores de movimiento, a los métodos biométricos para control de acceso, pasando por sistemas de cámaras de vigilancia de todo tipo, analógicas, digitales, rayos X, térmicas, etc.

Con los avances tecnológicos estos métodos han ido perfeccionándose y haciéndose cada vez más precisos, y en ningún momento se duda de la efectividad de estos, pero ahí donde acaban las posibilidades de estos sistemas es donde este proyecto cobra sentido. Se pretende pasar a un siguiente nivel de seguridad, en el que se analicen las ondas que rodean al sistema para mantener la seguridad ante posibles amenazas que puedan pasar desapercibidas al resto de sistemas.

Existen multitud de proyectos que realizan escaneos de este tipo de paquetes. Entre los más conocidos se encuentra el proyecto Aircrack-ng[3] que dispone de su herramienta Airodump-ng para escanear todo tipo de información de redes próximas. Este es un proyecto multiplataforma (Linux, Windows) que tiene comienzo en 2006, y pensado para la realización de auditorías a redes inalámbricas. No se ha encontrado ningún proyecto en la actualidad que realice las funciones que se pretenden desarrollar.

[3] AIRCRACK-NG https://www.aircrack-ng.org/

El Sistema Alejandra, complementa esos proyectos de escaneos ya existentes con una funcionalidad de inteligencia sobre la información obtenida. De esta manera se pretende poder baremar riesgos basados en localizaciones anteriores de los dispositivos encontrados, para que un sistema autónomo de inteligencia artificial lo valore y junto con el resto de las características del dispositivo sea capaz de determinar si se trata de una posible amenaza real.

2.2 Redes Inalámbricas

La tecnología WIFI surge debido a la necesidad de establecer una comunicación inalámbrica entre diferentes tipos de dispositivos. Es por ello que en el año 2000 las empresas 3Com, Airones, Intersil, Lucent Technologies, Nokia y Symbol Technologies forman la WECA (*Wireless Ethernet Compatibility Alliance*), en la actualidad llamada *WIFI Alliance*, crean la marca WI-FI[4], la cual, cumpliendo con el estándar 802.11b diseñado por el *Institute of Electrical and Electronics Engineers*, en adelante IEEE, garantiza que los diferentes dispositivos que se encuentren bajo la citada marca serán compatibles entre sí para realizar comunicaciones inalámbricas.

2.2.1 Estándar 802.11

Este estándar marca la guía común con la que se inicia la tecnología WIFI como hemos comentado anteriormente. Este estándar opera tanto en la banda 2,4 GHz como en la de 5 GHz, dependiendo del protocolo que utiliza como se detallará más adelante. Este punto es importante para el proyecto, ya que es un dato a tener en cuenta para la escucha de los diferentes canales existentes. El sistema debería barrer todos los canales, incluso los rangos que operen en otras regiones definidas por la Unión Internacional de Telecomunicaciones, en adelante UIT.

Cada región tiene una regulación diferente de sus frecuencias (ver Figura 10), llegando a ser más restrictivas incluso dentro de cada país o zona en algunas ocasiones.

[4] WIFI ALLIANCE. https://www.wi-fi.org/

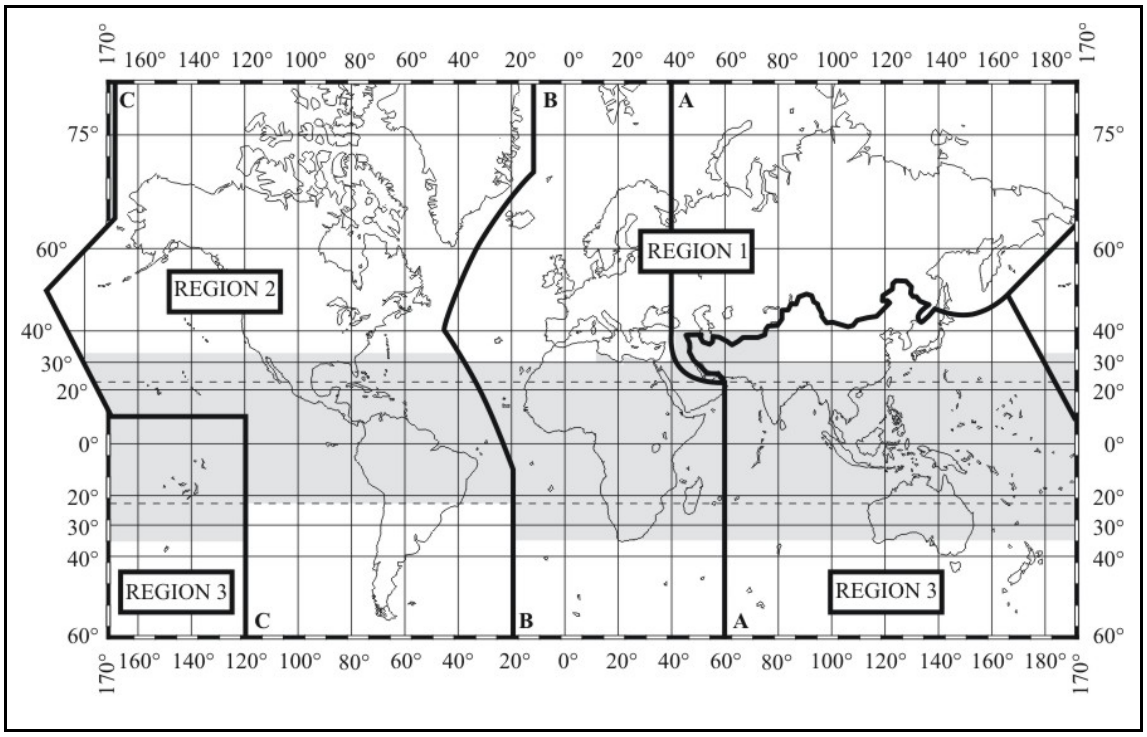


Figura 10 - Regiones definidas por la UIT[5]

Así pues, en lo que atañe a la frecuencia de 2,4 GHz, se define para la región que incluye Europa y en la que está incluida España un rango que va desde los 2,4 GHz hasta los 2,4835 GHz (ver Figura 11), y se estipula una potencia máxima de emisión para este tipo de dispositivos de 100mW.

Si se analiza ese espectro de frecuencias, se obtiene una división por canales que se corresponde con el siguiente gráfico:

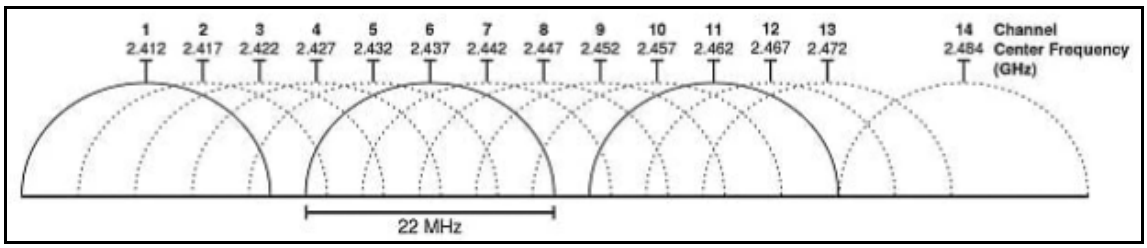


Figura 11 - Espectro de canales banda 2.4 GHz[6]

[5] ITU www.itu.int
[6] NETSPOT www.netspotapp.com/es/wifi

Se puede apreciar un total de 13 canales válidos en la franja europea y otro más que es el canal 14 que solo está permitido en la región 3. A simple vista también se aprecia que existe una superposición de canales ya que el primer canal tiene como frecuencia central 2412 MHz y un ancho de banda de 22 MHz, continuándole el canal 2 con una diferencia de frecuencia central de 5 MHz y el mismo ancho de banda. Esto va ocurriendo con el resto de los canales a lo largo de toda la banda. (ver Tabla 1)[7]

Canal	Frecuencia inferior	Frecuencia central	Frecuencia superior
1	2401	2412	2423
2	2406	2417	2428
3	2411	2422	2433
4	2416	2427	2438
5	2421	2432	2443
6	2426	2437	2448
7	2431	2442	2453
8	2436	2447	2458
9	2441	2452	2463
10	2446	2457	2468
11	2451	2462	2473
12	2456	2467	2478
13	2461	2472	2483
14	2473	2484	2495

Tabla 1 - Espectro de frecuencias de canales en 2,4 GHz

Podemos ver que los canales se superponen generando interferencias, salvo en el caso del canal 14 que se encuentra desplazado y genera mucha menos interferencia.

En lo referente a la banda de 5GHz este problema desaparece ya que los canales se encuentran desplazados para evitar interferencias. En este caso existen 25 canales de 20 MHz cada uno que no se superponen (ver Tabla 2)[8].

[7] NETSPOT https://www.netspotapp.com/es/wifi-channel-scanner.html
[8] NETSPOT https://www.netspotapp.com/es/wifi-channel-scanner.html

Ancho de Banda	Canales
20 MHz	36, 40, 44, 48, 52, 56, 60, 64, 100, 104, 108, 112, 116, 120, 124, 128, 132, 136, 140, 144, 149, 153, 161, 165, 169
40 MHz	38, 46, 54, 62, 102, 110, 118, 126, 134, 142, 151, 159
80 MHz	42, 58, 106, 122, 138, 155
160 MHz	50, 114

Tabla 2 - Espectro de canales banda 5 GHz

A nivel arquitectura el estándar 802.11 queda encuadrado en el modelo OSI en sus capas uno y dos, esto es la capa física y de enlace de datos (ver Figura 12).

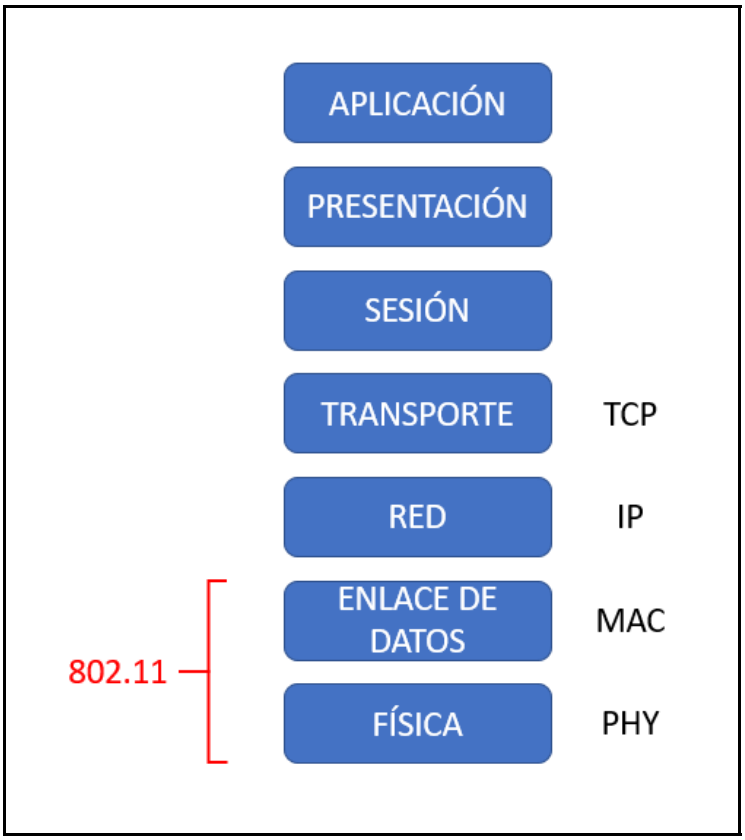


Figura 12 - Estándar 802.11 en el modelo OSI

Si se entra en detalle, se aprecia que cada capa involucrada en el estándar se puede dividir en dos subcapas (ver Figura 13).

En primer lugar, la capa física se puede dividir en la subcapa *Physical Media Dependent* donde el estándar define los sistemas de transmisión a nivel físico, en este caso Infrarrojos, FHSS, DSSS y OFDM. Así como en la subcapa

Physical Layer Convergence Procedure, la cual adapta los sistemas anteriores con la capa de enlace de datos.

Por otro lado, en la capa de enlace de datos tenemos la subcapa *Media Access Control*, en adelante MAC, que donde se especifica el protocolo de acceso, así como otras características que serán útiles para el desarrollo de este proyecto, como es la fragmentación de tramas de los datos transmitidos. La otra subcapa es la *Logical Link Control*, la cual es común a todos los estándares 802, y que en este caso carece de mayor interés.

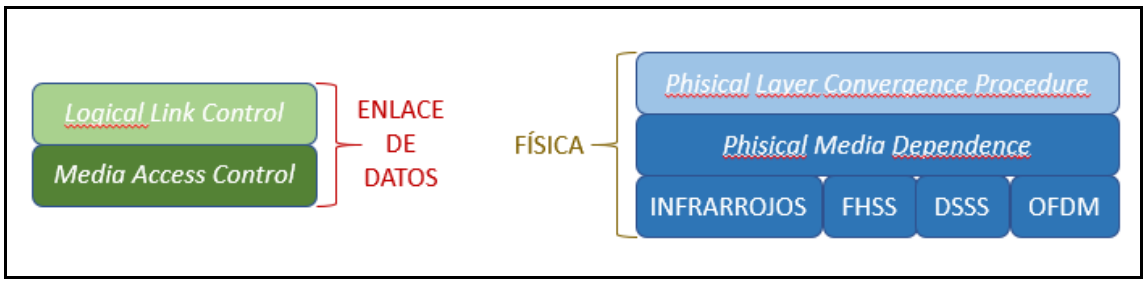


Figura 13 - Desglose de subcapas física y de enlace

En cuanto a sus protocolos, el estándar 802.11 fue creado por el IEEE en el año 1997 definiendo las características físicas y el control del acceso al medio. En concreto dispone de cuatro subcapas dentro de la PMD para la transmisión de datos, tres de radiofrecuencia, una de ellas empleando la técnica de espectro ensanchado por salto de frecuencia (FHSS), la otra de espectro ensanchado de secuencia directa (DSSS) y otra de multiplexación por división de frecuencias ortogonales (OFDM). La cuarta es de infrarrojos de banda base.

En adelante se han ido desarrollando distintos protocolos dentro del estándar 802.11 para adecuar y mejorar las condiciones de conexión. Concretamente esta norma es revisada en 1999 y posteriormente mediante los siguientes protocolos[9]:

Protocolo	Descripción
802.11b	Esta versión fue la primera tras la original y como características tiene una velocidad de transmisión de 11 Mbps a través de la banda de frecuencia de 2,4 GHz.
802.11a	Este estándar también se definió en 1999, y es capaz de aumentar la velocidad hasta los 54 Mbps en la banda de frecuencia de 5 GHz. Este protocolo dispone de 8 canales de red inalámbrica y 4 reservados para las conexiones punto a punto. El desarrollo de este estándar surge con la necesidad de disminuir las interferencias en la banda 2,4 GHz debidas a la saturación de esta. Aunque como contrapartida, se trata de una señal con menos alcance debido a que por su longitud de onda se ve más afectada siendo absorbida por los obstáculos o paredes.
802.11c	Se trata de un protocolo diseñado para comunicaciones de dos redes distintas o de diferente tipo. Otorga una velocidad teórica de unos 600Mbps y acepta tanto la banda de 2,4 GHz como la de 5GHz
802.11d	Este protocolo únicamente permite el uso del estándar de manera internacional. Estandariza los diferentes rangos disponibles limitados por el origen del dispositivo para que puedan intercambiarse información
802.11e	Esta especificación optimiza la conexión en entornos públicos, residenciales y de negocios, introduciendo características QoS y multimedia.
802.11f	Se trata de una ampliación que usa el protocolo IAPP que permite a un usuario cambiar de punto de acceso sin perder conexión.
802.11g	Este protocolo es la evolución del 802.11b. Al igual que el 802.11b utiliza la banda de 2,4 GHZ, aunque es capaz de llegar a una velocidad de transmisión de 54 Mbps.
802.11h	Se trata de un protocolo diseñado para resolver problemas de coexistencia entre dispositivos WIFI con sistemas de radares y satélites.
802.11i	Se implementa la WPA2 en redes inalámbricas y se diseña para aumentar la seguridad en los protocolos de autenticación y codificación.
802.11j	Se trata de un protocolo similar al 802.11h, pero para sistemas de Japón.

[9] IEEE https://www.ieee.org/

802.11k	Este protocolo se diseñó para mejorar la gestión de los clientes de una red calculando desde los puntos de acceso los recursos de radiofrecuencia de los dispositivos que se encuentran conectados.
802.11n	Este protocolo se diseñó en 2004 y se trata de una gran ampliación que permite la coexistencia de la banda de 2,4 GHz con la de 5 GHz, haciendo uso simultáneo de ambas. Con este protocolo se alcanzan velocidades teóricas de hasta 600 Mbps.
802.11p	Se trata de un estándar diseñado para comunicaciones de automóviles y utiliza la banda de 5,9 GHz y 6,2 GHz.
802.11r	Aumenta la velocidad en los cambios de conexión entre diferentes nodos consiguiendo que sea más estable.
802.11v	Se utiliza para la configuración remota de los dispositivos conectados a la red.
802.11w	Se trata de otro protocolo de ampliación de seguridad para redes inalámbricas.
802.11ac	Este protocolo es una mejora del 802.11n que permite velocidades teóricas del orden de 1.3 Gbps dentro de la banda de 5GHz.
802.11ax	Se trata de la última versión del estándar 802.11 y se conoce comúnmente como WIFI6. Este protocolo ha ampliado el espectro hasta los 6 GHz. También puede operar en los espectros de 2,4 y 5 GHz.

Tabla 3 - Tabla de Protocolos 802.11

Pese a las últimas novedades existentes durante el presente año que hacen referencia al WIFI6, este espectro queda fuera del alcance de este proyecto. En primer lugar, porque la cantidad de dispositivos que ocupan este espectro a nivel usuario ahora mismo es mínimo, además de la dificultad de conseguir equipo para poder desarrollarlo.

2.2.2 Composición de tramas inalámbricas

Para este proyecto este es uno de los puntos clave y básicos a analizar, ya que de ellas se obtendrán los datos que nos interesan para conseguir los objetivos. Existen tres tipos de tramas: las de gestión, las de control y las de datos.

- Trama de gestión: Como su nombre indica, realiza la gestión de la conexión, autenticación y desautenticación, descubrimiento de redes conocidas, etc…
- Trama de control: Se encarga de que la conexión funcione correctamente, realiza las confirmaciones, reservas de canal, etc.
- Trama de datos: Transportan la información de las capas superiores.

En el marco de este proyecto se centrará la atención sobre las tramas de gestión, concretamente sobre un subtipo denominado *Probe Request*. Estas tramas se encargan de hacer las funciones de sonda, emitiendo peticiones de continuamente de las redes conocidas del dispositivo, de esta manera en el momento que encuentra una red conocida, la red le responde mediante una trama de tipo *Probe Response* y comienza el proceso de autenticación.

Una trama de tipo *Probe Request*, está formada de la siguiente manera:

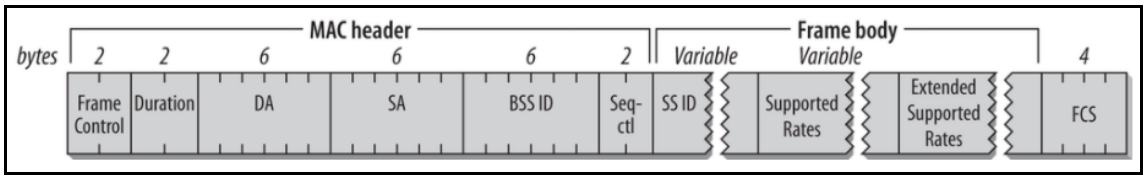


Figura 14 - Trama *Probe Request* [10]

Se puede apreciar que su cuerpo contiene dos campos, uno de ellos contiene el SSID que el dispositivo está buscando, y el otro (dividido en dos) contiene características soportadas por el dispositivo móvil. Esta trama es la misma que

[10] OREALLY https://www.oreilly.com/library/view/80211-wireless-networks/0596100523/ch04.html

se utiliza para descubrir puntos de acceso disponibles, solo que, en este caso, el campo SSID será “0”.

Para detallar un poco más esto último, se puede ver que la cabecera se encuentra dividida en varios campos de los que se obtendrá parte de la información útil para el sistema.

- *Frame Control*: Número de control de trama.
- *Duration*: Campo de control de tiempo de la trama.
- DA (*Destination Address)*: Este campo contiene la dirección física (MAC) de destino a la que se dirige la trama, normalmente en este tipo de tramas la dirección suele ser Broadcast (ff:ff:ff:ff:ff:ff).
- SA (*Destination Address)*: Este campo contiene la información de la dirección MAC del dispositivo que realiza la petición.
- BSSID: En este campo se refleja la dirección MAC del dispositivo AP (*Access Point)*. En el caso de interés para el proyecto, este campo debe ser igual al de la dirección de destino (DA) ya que son pruebas de sus redes conocidas.
- Seq-ctl: Representa el número de secuencia de la trama.

De la cabecera se puede extraer como dato de interés la dirección MAC fuente, que le servirá al sistema para realizar el primer cribado por una lista de dispositivos de confianza para saber si se trata de un dispositivo autorizado o no.

El resto de datos, como el DA o el BSSID deben ser en broadcast, es decir, su valor debería ser ff:ff:ff:ff:ff:ff ya que el dispositivo no se encuentra vinculado a ninguna red.

El resto de los datos 3de interés para el proyecto se obtienen del cuerpo de la trama, donde encontramos:

- SSID: Valor en el que se refleja el SSID de la red conocida que el dispositivo está buscando.
- *Supported Rates*: Envía las tasas de transferencia de datos soportadas por el dispositivo.

De aquí se obtiene el/los SSID que el dispositivo está intentando localizar y sobre el que se hará posteriormente el estudio.

Visto esto, se puede entender que estos paquetes son el inicio en un proceso de asociación entre un dispositivo y una estación. A continuación, se presenta el proceso completo de asociación de un dispositivo, en el que vemos que primero se encuentra la búsqueda de redes, posteriormente las autenticaciones y por último la asociación (ver Figura15).

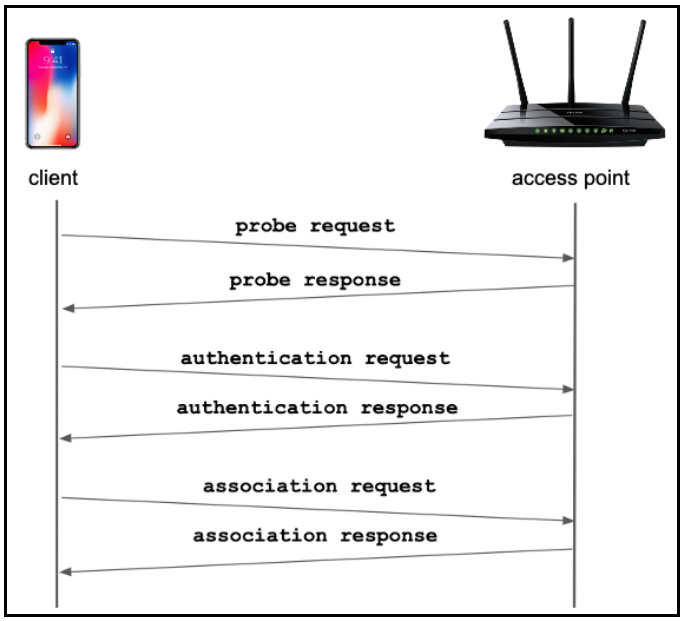


Figura 15 - Proceso de asociación de un dispositivo y una estación[11]

[11] ICHIPRO https://ichi.pro/es/escaneo-de-dispositivos-moviles-a-traves-de-wi-fi-usando-pi-zero-w-141397806599198

2.2.2 Identificador Único de Organización (OUI)

En las direcciones físicas de los dispositivos, en adelante MACs, se puede encontrar el **identificador único de organización**[12]. Se trata de un número formado por tres grupos codificados en hexadecimal que ocupan los tres primeros campos dentro de la MAC.

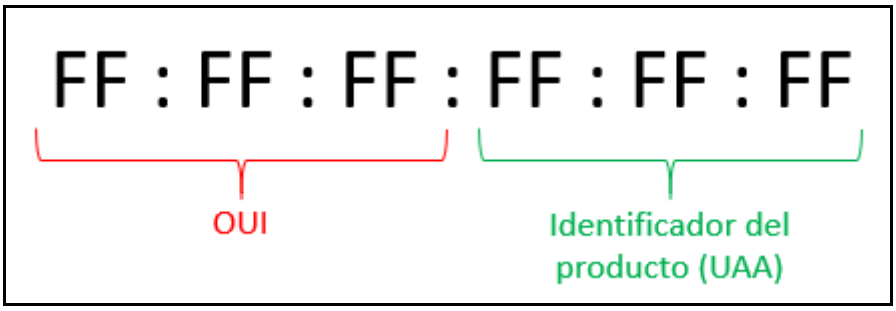


Figura 16 - Dirección MAC

Esta codificación es un identificador del fabricante del dispositivo, que es asignado y regulado por el IEEE. Teóricamente, cada dispositivo dispone de su propia MAC. Se dice teóricamente porque este número es susceptible de ser falseado. Y es ahí donde este apartado cobra importancia para el proyecto. Este OUI, será objeto de análisis junto otras variables a lo largo de la implementación SMART del mismo, donde un algoritmo de inteligencia artificial, en función de diferentes valores determinará la posibilidad de que sea real o falso.

2.3 Inteligencia Artificial

Aunque el desarrollo de sistemas de Inteligencia Artificial[13] puede resultar muy complejo, entender en líneas generales como funciona esta tecnología no lo es tanto.

---

[12] OREALLY https://www.oreilly.com/library/view/80211ac-a-survival/9781449357702/ch03.html
[13] GÉRON, Aurélien. Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems. https://es1lib.org/book/5207527/b6e3a9

En primer lugar, podríamos definir lo que se considera inteligencia artificial como un sistema que es capaz de dar la solución a un problema o situación razonando de una manera similar a la de un ser humano. Son sistemas creados para imitar el razonamiento humano.

Dentro de los complejos funcionamientos de esta tecnología, se encuentra el *Machine Learning*, donde los seres humanos entrenan sistemas para que sean capaces de reconocer patrones. Se le proporcionan datos que describen la problemática y el propio sistema realiza un aprendizaje sobre ello. Como ejemplo, se le podría aportar a un sistema datos de cómo es un coche, tales como formas y distribución de ventanillas, proporciones, ruedas, etc… y posteriormente entrenarlo mandándole fotografías de coches. Una vez el sistema aprenda, sería capaz de reconocer imágenes de coches con una cierta probabilidad.

También se puede detallar que existen tres tipos de aprendizajes relacionados con el *Machine Learning*, estos son:

- Aprendizaje supervisado, en el que el sistema aprende mediante ejemplos etiquetados, dispone de *labels*. En este método de aprendizaje se le introducen datos con sus respectivas salidas, y la máquina aprende en base a intentar clasificar los mismos. Se trata de un método enfocado en la minimización de una función de coste.

- Aprendizaje no supervisado, en el que se proporcionan al sistema datos sin claves de respuesta y es el sistema solo el que trata de clasificarlo y encontrar patrones adecuados. Cuantos más datos se vayan aportando al sistema, mejor respuesta se obtendrá.

- Aprendizaje reforzado, en el que se proporcionan al sistema entornos simulados. El sistema recibe datos y trata de enseñar a un agente estudiando el entorno, tomando decisiones, posteriormente observa los resultados y aprende de ello. Perfeccionando lo máximo posible el sistema.

Estos modelos de aprendizaje siempre disponen de algoritmos que realizan las tareas del sistema. Los más populares son:

- Algoritmos de regresión lineal, en el que se estudian relaciones entre las variables de las que una es dependiente y el resto son cambiantes.

- Algoritmos bayesianos, que se tratan de algoritmos de clasificación basados en el teorema de Bayes clasificando cada valor de manera independiente a cualquier otro.

- Algoritmos de agrupación, que sirven para clasificar datos no etiquetados. Este algoritmo busca grupos comunes dentro de los datos disponibles con un numero de grupos representados mediante una variable k.

- Algoritmos basados en árbol de decisión, que representan sus salidas a modo de diagrama de flujo mostrando las posibilidades de cada bifurcación.

- Algoritmos de redes neuronales, construidas mediante capas de unidades simulando un cerebro biológico.

- Algoritmos de reducción de dimensión, que como su nombre indica reducen el número de variables para encontrar la información.

- Algoritmos de *Deep Learning*, en el que se le proporcionan multitud de datos que cumplan las condiciones de lo que debe aprender y él extrae los patrones necesarios para la resolución del problema. Este tipo de algoritmos están basados en redes neuronales formadas por multitud de capas, lo que permite identificar patrones mucho más complejos que el resto de algoritmos.

En este caso, el concepto de *Deep Learning* es demasiado complejo para abordarlo, son necesarios sistemas que hacen las funciones de redes neuronales, etc… que se salen totalmente del alcance y propósito del proyecto. Sin embargo, el concepto de *Machine Learning*, cubre las necesidades de este,

en el que un algoritmo realizará un estudio de patrones de direcciones MAC legítimas junto los datos de sus redes conocidas y se intentará localizar con el mismo, amenazas que puedan aparecer en el sistema discriminando falsos positivos.

3. Composición del proyecto

3.1 Hardware

Para la realización del proyecto se van a utilizar dos elementos fundamentales, en primer lugar, una Raspberry PI 4 B (ver figura 17) que hará las funciones de servidor, alojando el software desarrollado, y por otro lado una tarjeta de red ALFA AWUS1900 (ver figura 18) con la que se podrá realizar un inyectado de paquetes para completar las simulaciones.

- Raspberry PI 4 B: Se trata de una minicomputadora de placa única, en este caso con 8 GB de RAM y es ideal para este tipo de proyectos. Es ligera, acepta dispositivos externos mediante diferente tipo de conectores y con un nivel de computación más que suficiente para los cálculos que se requieren.

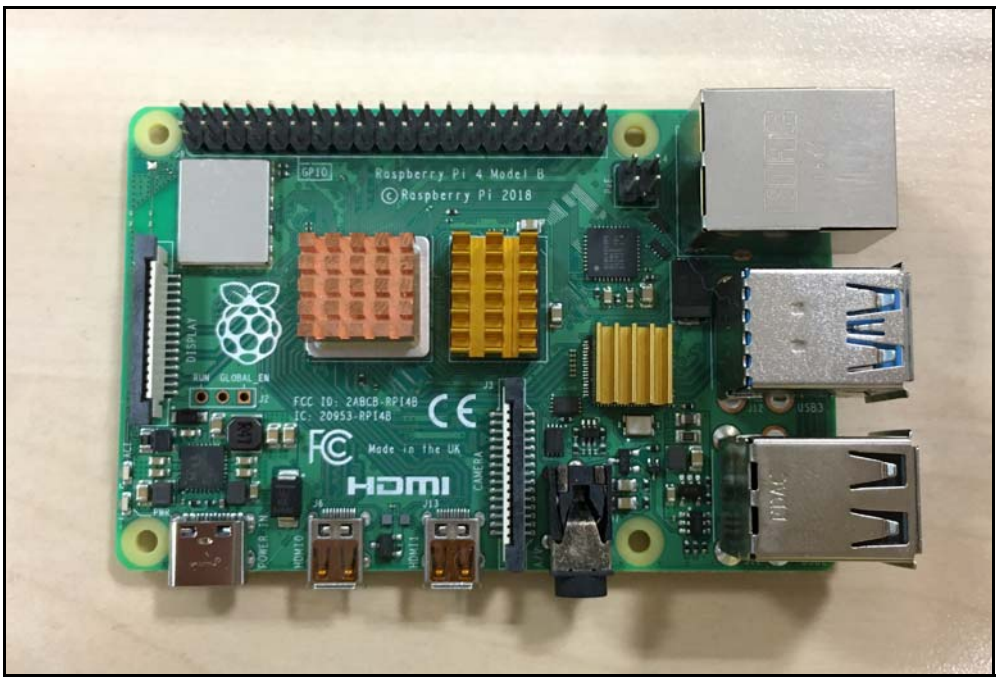


Figura 17 - Raspberry PI 4 B

Dispone de una tarjeta Wifi que será la que se utilice para monitorear la red mientras se inyectan paquetes para la simulación que se realizará con la tarjeta de red que se detalla posteriormente. Las especificaciones del dispositivo son:

- Cpu ARM CORTEX de 4 núcleos a 1.5 GHz
- Gráficos VideoCore VI
- Decodificador HEVC 4kp60
- Puerto Ethernet
- 2 puertos USB 3.0 y 2 puertos USB 2.0
- Dos puertos micro HDMI
- Un puerto USB-C para alimentación
- 8 GB DDR4
- Tarjeta WIFI

- Tarjeta de red ALFA AWUS1900: Este dispositivo es uno de los más completos del mercado. Soporta la tecnología 802.11 ac (aparte soporta también el a, n, g, b) con ratios de transferencia de más de 1300 Mbps en la banda de 5 GHz. También soporta la banda e 2,4 GHz con ratios de transferencia de 600 Mbps. Además, soporta multiplataforma por lo que permite trabajar tanto en Windows, como Mac, como el sistema operativo que usará el proyecto que es Linux.

Además, dispone de 4 antenas de 5Bdi, todas ellas de banda dual que permite grandes distancias de conexión.

Por último, cabe reseñar que dispone del procesador RTL8814U, que permite la inyección de paquetes *wireless,* lo que va a permitir realizar los diferentes testeos del sistema.

Figura 18 - Tarjeta de red ALFA AWUS1900

3.2 Software

A continuación, se detallarán las principales características software que se utilizará para el desarrollo del sistema.

En el dispositivo Raspberry PI se va a instalar un SO ágil de base Linux como la distribución Ubuntu que es base Debian y dispone de gran versatilidad. En el sistema operativo se instalará el entorno de programación Python en su versión 3.8 para el desarrollo del programa.

Python es un lenguaje de programación de alto nivel basado en objetos, muy dinámico y que permite el tratamiento de paquetes mediante librerías ya predefinidas. En la actualidad, numerosas empresas utilizan este lenguaje para sus sistemas *backend* por su fiabilidad y facilidad de desarrollo. En nuestro caso utilizaremos una librería común para realizar solicitudes, como es la librería Requests.

REQUESTS, permite realizar solicitudes de tipo GET, POST… etc, y manipularlas, a la vez que tratar las respuestas que se reciben, lo que será muy útil para la parte de análisis de datos del desarrollo, donde se harán

peticiones a la plataforma WIGGLE aportando las SSIDs y recibiendo un archivo de tipo JSON del que se obtendrán los datos de coordenadas de la ubicación del SSID conocido por el dispositivo.

Otra librería fundamental que se utilizará en el desarrollo es SCAPY, la cual se encarga de las funciones para la detección de paquetes, la identificación de estos y su tratamiento.

En lo referente al almacenamiento de datos, se hará mediante una base de datos de tipo SQLITE, tratándola mediante la librería SQLITE3. La elección de este tipo de bases de datos es por su facilidad de manejo y dinamismo, sin necesidad de tener servicios paralelos de gestión para su gestión como pasa por ejemplo en MYSQL que es una de las más conocidas.

Por último, para la implementación SMART, utilizaremos la librería SKLEARN desarrollada en Python y que permite el tratamiento de datos mediante el algoritmo ***k-nearest neighbors*****,** en adelante KNN, el cual se detallará posteriormente. Además, para las pruebas de concepto se utilizará la aplicación gratuita WEKA, con la que se podrá apreciar las características de precisión del algoritmo.

4. Sistema base (ALEJANDRA v.1.0)

4.1 Introducción

Para el desarrollo del código, como se ha detallado anteriormente se utilizará el lenguaje de programación PYTHON en su versión 3.x. junto con el lenguaje SQL para el manejo del contenido de las bases de datos. Se realizará de manera segmentada en 5 módulos diferentes denominados: alejandra.py, análisis.py, procesado.py, sniffer.py y comunicación.py. Adjunto igualmente se proporciona un archivo readme.md con las instrucciones de uso, así como los requisitos para su funcionamiento. **Se adjunta el código del programa en el ANEXO II**. El algoritmo del código es el siguiente:

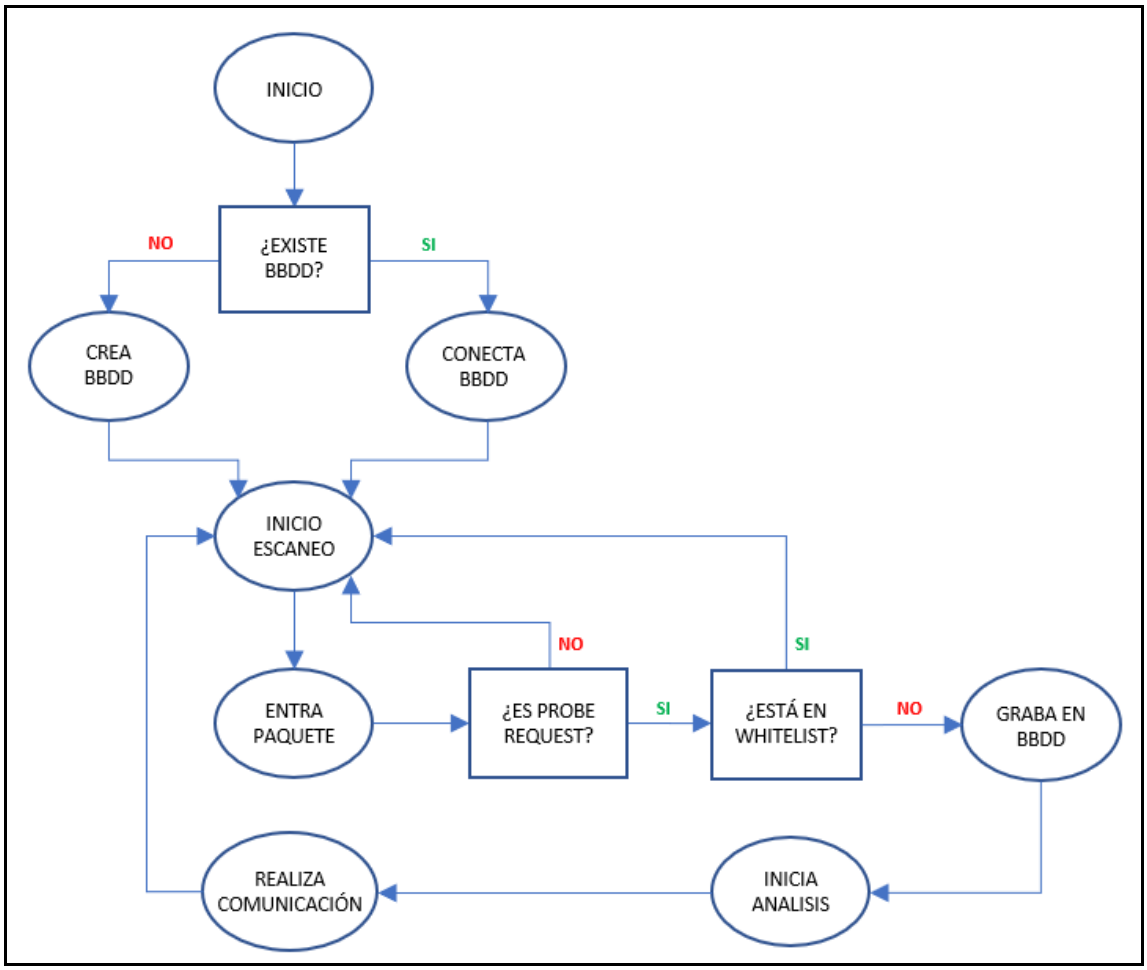


Figura 19 - Diagrama de flujo del sistema

4.2 Funcionamiento de los módulos

El programa es ejecutado mediante el archivo *alejandra.py*. En este punto es el único en el que el usuario interactuará con el sistema, y lo hará para decidir si quiere generar una base de datos nueva, o utilizar ya una existente a través de llamadas al módulo *procesado.py*. También exige que el usuario seleccione una interfaz de red para capturar paquetes. El propio sistema establecerá el modo monitor a la tarjeta de red y una vez realizado comenzará a escanear (ver Figura 20).

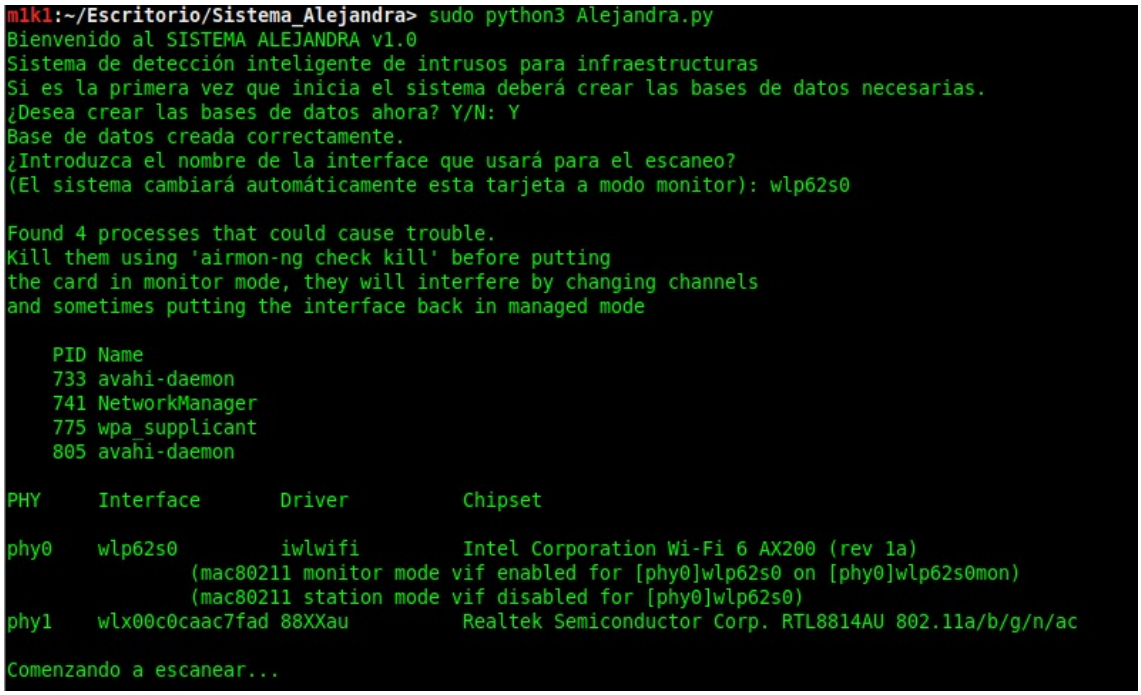


Figura 20 - Inicio del programa

La base de datos en esta primera versión del programa (sin sistema SMART activo) se almacenará en el archivo *BBDD_Wifishield.db*, y se compondrá de dos tablas, una denominada *Whitelist* y otra Amenazas. En la primera se almacenarán las direcciones MAC de los dispositivos permitidos y en la segunda se irán almacenando en dos columnas las direcciones MAC y las redes conocidas de los dispositivos que estos están buscando mediante la emisión de los paquetes *Probe Request*.

Durante la operación de escaneo, que se realiza a través del módulo *Sniffer.*py, el programa irá desechando paquetes que no sean de tipo *Probe Request*, y

cuando encuentra uno, evalúa si esa MAC se encuentra en la *Whitelist*, de no encontrarse, se iniciará un proceso de alarma que enviará los datos de la MAC y el SSID de la red conocida asociada al módulo de análisis, *análisis.py*. (ver figura 21)

Figura 21 - Escaneo y localización de amenaza

En este módulo se realizará una petición a una plataforma llamada WIGLE, de donde obtendremos datos de ubicación de esta red y de adquisición de esta información, que serán tratados y enviados al módulo *comunicación.py*, donde finalmente se preparará un email y es remitido al responsable de seguridad (ver

figura 22). La solicitud de información a la plataforma WIGLE se realiza utilizando la API de esta, y para ello se ha creado con carácter exclusivamente académico una cuenta con los siguientes datos de conexión a través de la API:

- Usuario= 'AID768a3e53108648827e3a2ae1220acc87'
- Password= '846f4909cbbd03ba73302c51b1b5c248'

Igualmente se ha creado la cuenta de correo *wifishieldproject@gmail.com* para la emisión de los emails haciendo las veces de la cuenta de correo de la empresa.


De wifishieldproject@gmail.com
Asunto WIFISHIELD, ALERTA DE SEGURIDAD
A mí <miguelangel.mcastilla@gmail.com>

ALERTA DE SEGURIDAD!! DISPOSITIVO POTENCIALMENTE PELIGROSO EN LAS INMEDIACIONES DEL COMPLEJO.

Dispositivo: b4:8b:19:94:65:53
SSID conocida: H4CK_4_Fun

Resultado 1:
Fecha: 2017-11-13T13:00:00.000Z
Pais: ES
Región: Comunidad de Madrid
Cuidad: Madrid
Calle: Calle de Aragón
Coordenadas: 40.44960022 -3.58063626
Link: https://www.google.es/maps/place/40.44960022%20-3.58063626

Resultado 2:
Fecha: 2020-08-14T12:00:00.000Z
Pais: ES
Región: Comunidad de Madrid
Cuidad: Madrid
Calle: Avenida de Aragón
Coordenadas: 40.44961929 -3.58088446
Link: https://www.google.es/maps/place/40.44961929%20-3.58088446

Resultado 3:
Fecha: 2016-03-31T14:00:00.000Z
Pais: ES
Región: Comunidad de Madrid
Cuidad: Madrid
Calle: None
Coordenadas: 40.44966125 -3.58130693
Link: https://www.google.es/maps/place/40.44966125%20-3.58130693

Figura 22 - Email de prueba enviado al responsable de seguridad

Cabe reseñar, que todo este proceso sería realizado por completo en segundo plano, pero se ha implementado impresiones en el código para que muestre su funcionamiento interno y su valoración académica.

En el email anterior se puede apreciar varios resultados, esto es porque se ha decidido obtener el historial completo de la plataforma con el fin de valorar de manera más optima la amenaza.

Para el ejemplo de funcionamiento anterior se ha creado una *whitelist* a base de datos obtenidos escaneando una zona concreta y posteriormente se han inyectado paquetes con la MAC: b4:8b:19:94:65:53 y el SSID: H4CK_4_Fun.

5. Implementación SMART (ALEJANDRA v.1.1)

5.1 Introducción

En el marco del proyecto, se pretende agregar al mismo un algoritmo de *Machine Learning* que permita discernir dentro de los positivos los que se traten de una amenaza real de los que puedan tratarse de falsos positivos. Estos falsos positivos podrían venir por un cambio de teléfono móvil, Tablet, ordenador portátil, etc… de un miembro del personal autorizado no comunicado debidamente al departamento de seguridad.

Para realizar esto se ha optado por un algoritmo de tipo "*k nearest neighbors",* en adelante KNN[14]. Este algoritmo es de tipo supervisado, e ideal para *datasheets* de poca cantidad de datos. Es sencillo y se puede usar para clasificar datos o predecir. En nuestro caso lo utilizaremos para clasificar nuevas entradas.

Su funcionamiento es bastante sencillo, se basa en alimentar al sistema con un *datasheet* que comprenda la totalidad de datos de interés con el que se entrenará al sistema. En el caso que nos ocupa, transformar cada MAC de los dispositivos en un vector que refleje mediante ceros y unos que redes son conocidas para este dispositivo sobre el total de las redes registradas por el sistema. Una vez vectorizado cada dispositivo, se agrupan por clases, en el caso de este proyecto se agruparán por trabajadores. Es importante que los datos que se usan para entrenar al sistema sean balanceados, es decir, que haya la cantidad de datos más similar para cada clase.

Otro dato importante en el algoritmo es el valor de "k", este valor representa la cantidad de "vecinos" (por vecinos se entienden puntos cercanos en el espacio vectorial) que tendrá en cuenta para calcular la distancia con el dato introducido para su clasificación.

---

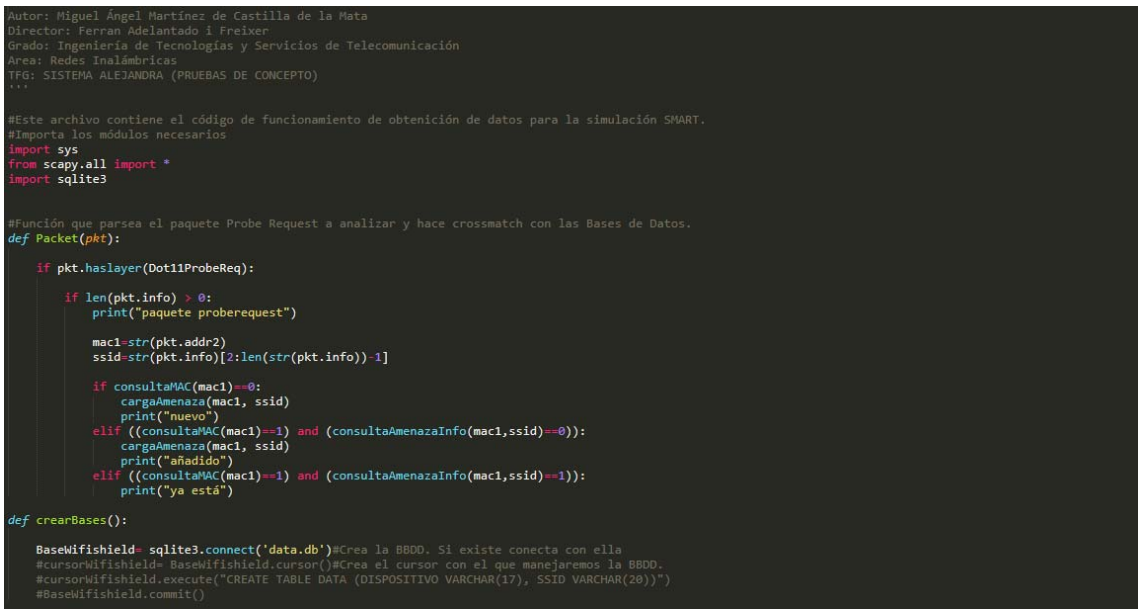
[14] WITTEN, Ian H. & FRANK Eibe (2005). Data Mining. Practical Machine Learning Tools and Techniques. San Francisco. Elsevier

Para terminar, hay que tener en cuenta que para entrenar el sistema se recomienda que junto con los datos de entrenamiento se introduzca una serie de datos que servirán de test del mismo. Lo óptimo es una proporción de inyección de datos del 70% datos de entrenamiento y 30% datos de testeo.

5.2 Preparación del entorno de simulación y pruebas de concepto

Una vez seleccionado el algoritmo a utilizar, en este caso KNN, y debido a la dificultad de obtener un entorno 100% real para la ejecución de este, se procede a obtener un entorno semi real para realizar las pruebas de concepto necesarias.

En primer lugar, se necesita modificar el modelo de tablas de la base de datos referenciada en el programa original. Para este supuesto se necesita una tabla que contenga a parte de los dispositivos autorizados, las redes conocidas de estos. Para ello se utilizará un script, denominado *sniff.py,* (ver figura 23 y 24) que se encargará de recoger datos de una zona concreta y almacenarlos en una base de datos acorde a las necesidades especificadas, concretamente “data.db”.



```
Autor: Miguel Ángel Martínez de Castilla de la Mata
Director: Ferran Adelantado i Freixer
Grado: Ingeniería de Tecnologías y Servicios de Telecomunicación
Area: Redes Inalámbricas
TFG: SISTEMA ALEJANDRA (PRUEBAS DE CONCEPTO)
'''

#Este archivo contiene el código de funcionamiento de obtención de datos para la simulación SMART.
#Importa los módulos necesarios
import sys
from scapy.all import *
import sqlite3


#Función que parsea el paquete Probe Request a analizar y hace crossmatch con las Bases de Datos.
def Packet(pkt):

    if pkt.haslayer(Dot11ProbeReq):

        if len(pkt.info) > 0:
            print("paquete proberequest")

            mac1=str(pkt.addr2)
            ssid=str(pkt.info)[2:len(str(pkt.info))-1]

            if consultaMAC(mac1)==0:
                cargaAmenaza(mac1, ssid)
                print("nuevo")
            elif ((consultaMAC(mac1)==1) and (consultaAmenazaInfo(mac1,ssid)==0)):
                cargaAmenaza(mac1, ssid)
                print("añadido")
            elif ((consultaMAC(mac1)==1) and (consultaAmenazaInfo(mac1,ssid)==1)):
                print("ya está")

def crearBases():

    BaseWifishield= sqlite3.connect('data.db')#Crea la BBDD. Si existe conecta con ella
    #cursorWifishield= BaseWifishield.cursor()#Crea el cursor con el que manejaremos la BBDD.
    #cursorWifishield.execute("CREATE TABLE DATA (DISPOSITIVO VARCHAR(17), SSID VARCHAR(20))")
    #BaseWifishield.commit()
```

Figura 23 - Código del script Sniff.py (Parte I)

```
def consultaMAC(mac):


    BaseWifishield= sqlite3.connect('data.db')
    cursorWifishield= BaseWifishield.cursor()
    cursorWifishield.execute("SELECT DISPOSITIVO FROM DATA")
    lista=cursorWifishield.fetchall()
    match=0
    for a in lista:
        if mac in a:

            match=1

    return match

def consultaAmenazaInfo(mac, info):

    BaseWifishield= sqlite3.connect('data.db')
    cursorWifishield= BaseWifishield.cursor()
    cursorWifishield.execute("SELECT SSID FROM DATA where DISPOSITIVO=?",(mac,))
    wifis=cursorWifishield.fetchall()
    match=0
    for a in wifis:
        if info in a:

            match=1
    return match

def cargaAmenaza(mac,info):
    BaseWifishield= sqlite3.connect('data.db')
    cursorWifishield= BaseWifishield.cursor()
    cursorWifishield.execute("INSERT INTO DATA (DISPOSITIVO,SSID) VALUES (?,?)",(mac,info))
    BaseWifishield.commit()

os.system("airmon-ng start wlp62s0")
crearBases()
sniff(iface="wlp62s0mon", count= 0,prn=Packet)
```

Figura 24 - Código del script Sniff.py (Parte II)

Posteriormente se crea un archivo que contiene el total de las redes conocidas y otro el total de los dispositivos. Estos archivos serán nombrados “SSIDs” y “DISPOSITIVOS” respectivamente y se utilizarán posteriormente en el script clasificación.py (ver figura 25). “*En entorno real esta manipulación de datos se haría de manera transparente mediante variables, pero como se trata de una simulación académica se ha optado por dejar todo visible en archivos para su comprobación y evaluación. También se han añadido funciones de impresión de datos en ejecución (ver figura 26) con el mismo objetivo.”* Este último script será el encargado de crear el archivo *vectores.csv* que contendrá el *datasheet* con el que se entrenará el algoritmo KNN.

```python
Autor: Miguel Ángel Martínez de Castilla de la Mata
Director: Ferran Adelantado i Freixer
Grado: Ingeniería de Tecnologías y Servicios de Telecomunicación
Area: Redes Inalámbricas
TFG: SISTEMA ALEJANDRA (PRUEBAS DE CONCEPTO)
'''

#Este archivo contiene el código de creación del datasheeet para la Inteligencia Artificial.

import sqlite3


datos= sqlite3.connect('data.db')#Crea la BBDD. Si existe conecta con ella
cursor= datos.cursor()#Crea el cursor con el que manejaremos la BBDD.
ssids=open('SSIDs', 'r')

a=[] #Lista que contendrá las redes conocidas
for i in ssids:
    a.append(i.replace("\n",""))

dispos=open('DISPOSITIVOS', 'r')
vectores=open('vectores.csv', 'a')

#Añade la cabecera al archivo que contendrá el datasheet
vectores.write(",".join(a)+"\n")

#Recorre los dispositivos y crea un vector para cada uno de ellos en el que marcará como 1 la posición en la que se encuentren sus redes conocidas
for i in dispos:
    mac=i.replace("\n","")
    #Lista con la MAC convertida a vector.
    b=["0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0"]
    c=cursor.execute("SELECT SSID FROM DATA where DISPOSITIVO=?",(mac,)).fetchall()


    for i in c:
        x=i[0]
        indice=a.index(x)
        b[indice]="1"

    vectores.write(",".join(b)+"\n")#Agrega una nueva línea al datasheet
```

Figura 25 - Código del script clasificacion.py

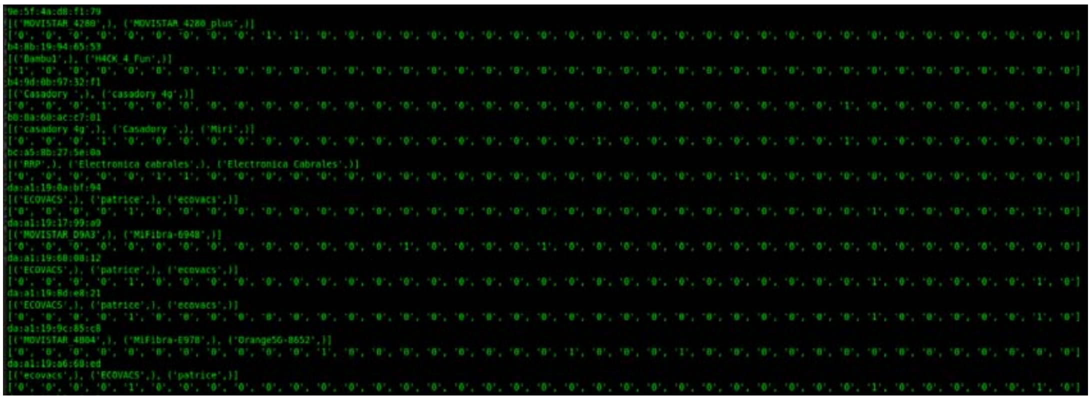


Figura 26 - Extracto de formación de vectores

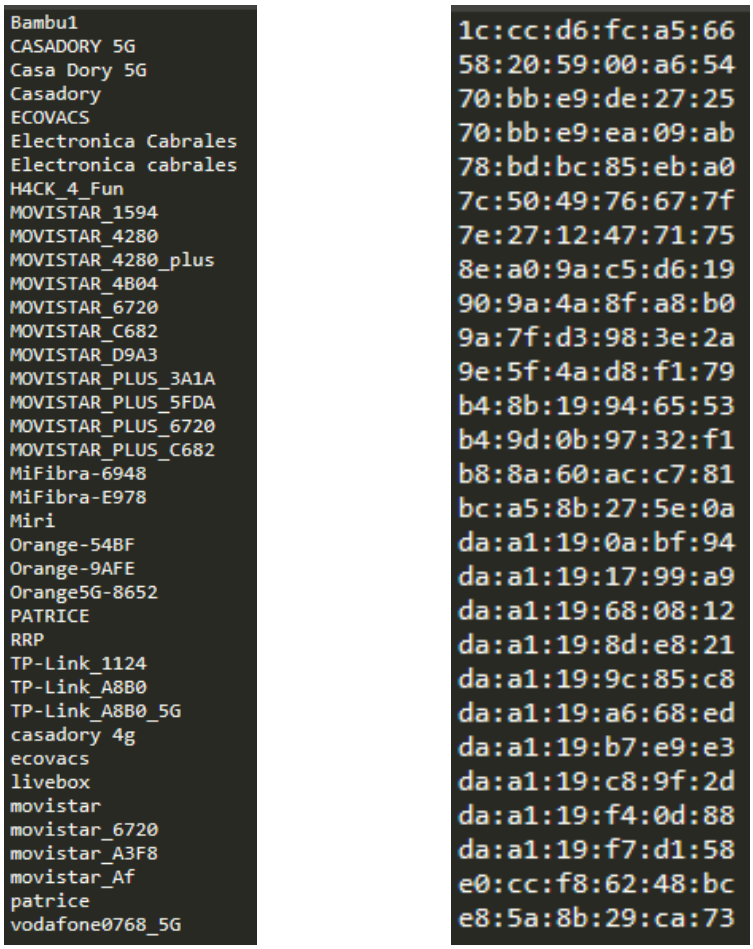
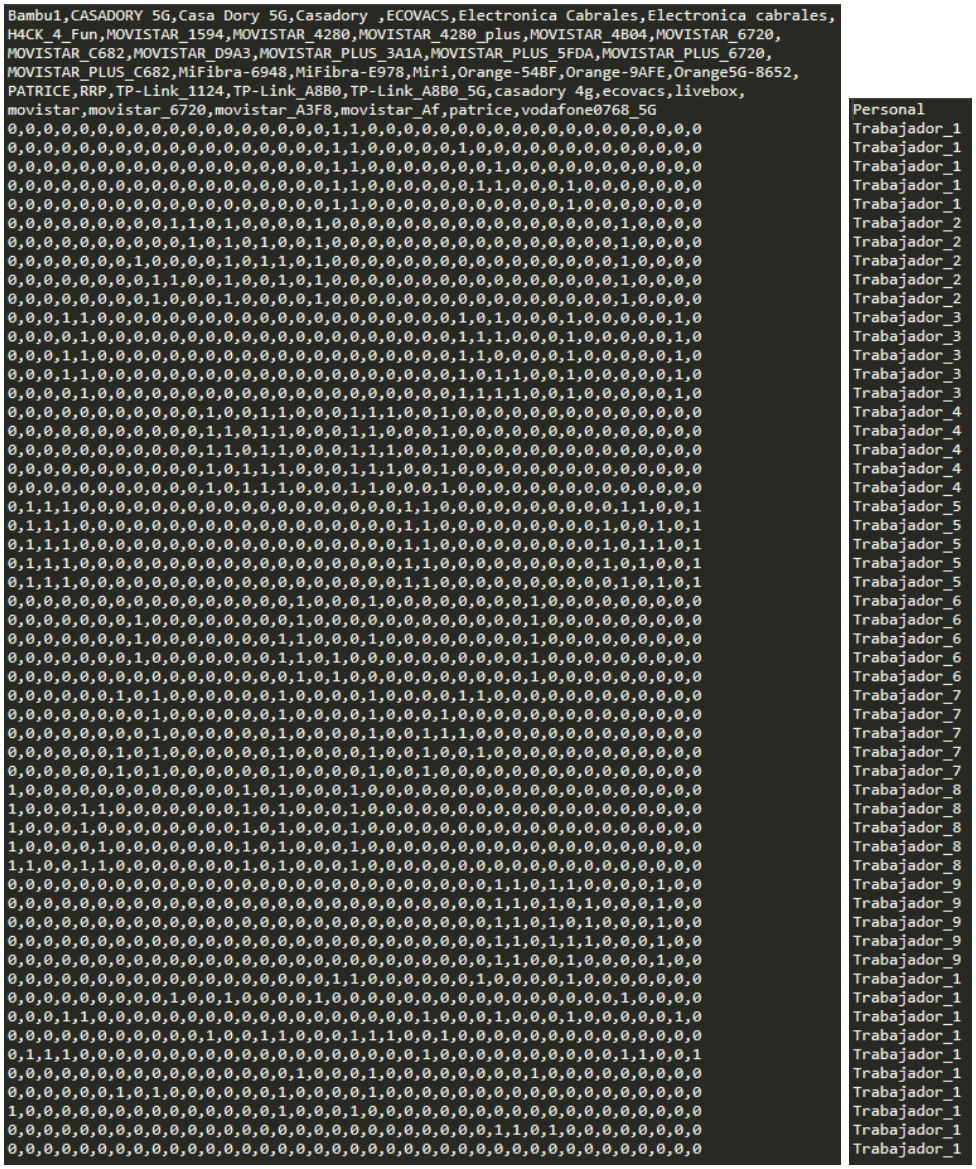


Figura 27 - Contenido de los archivos SSIDs y DISPOSITIVOS

Como se puede observar, se registra un total de 39 redes conocidas en el sistema y de 27 dispositivos que formarían la *whitelist*. De todo ello y tras tratarlos con el script clasificacion.py, se obtiene el *datasheet* que se utilizará en el algoritmo. En este caso será un archivo de formato "*.csv*" formado por 27 vectores, uno por dispositivo en el que se reflejarán 39 atributos, que serán ceros o unos dependiendo de la pertenencia o no a la red que corresponda.

En este punto es donde se tiene que intervenir sobre los datos, ampliando los mismos de manera artificial para poder tener un conjunto decente en cantidad para poder entrenar al sistema. Este conjunto de datos total (ver figura 29) será también segregado por trabajadores de manera artificial, esto se hace simulando que cada trabajador, en el caso que nos ocupa serán un total de 9 trabajadores, dispone de 5 dispositivos diferentes. Además, se introducirán un total de 10 vectores extra para realizar la fase de

test del sistema (en este caso las 10 últimas). Conjuntamente, al algoritmo hay que presentarle un dato que se denomina “etiqueta”, que será el dato que se use para clasificar cada vector, y donde luego el sistema encuadrará las predicciones sobre datos nuevos que le aportemos en este caso los trabajadores como aparecen en la figura 29. Cabe reseñar que los vectores de test hay que aportarle una etiqueta igualmente para ver la eficacia del sistema.



Bambu1,CASADORY 5G,Casa Dory 5G,Casadory ,ECOVACS,Electronica Cabrales,Electronica cabrales, H4CK_4_Fun,MOVISTAR_1594,MOVISTAR_4280,MOVISTAR_4280_plus,MOVISTAR_4B04,MOVISTAR_6720, MOVISTAR_C682,MOVISTAR_D9A3,MOVISTAR_PLUS_3A1A,MOVISTAR_PLUS_5FDA,MOVISTAR_PLUS_6720, MOVISTAR_PLUS_C682,MiFibra-6948,MiFibra-E978,Miri,Orange-54BF,Orange-9AFE,Orange5G-8652, PATRICE,RRP,TP-Link_1124,TP-Link_A8B0,TP-Link_A8B0_5G,casadory 4g,ecovacs,livebox, movistar,movistar_6720,movistar_A3F8,movistar_Af,patrice,vodafone0768_5G	Personal
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	Trabajador_1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0	Trabajador_1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0	Trabajador_1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,1,1,0,0,0,0,1,0,0,0,0,0,0	Trabajador_1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0	Trabajador_1
0,0,0,0,0,0,0,0,0,1,1,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0	Trabajador_2
0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0	Trabajador_2
0,0,0,0,0,0,0,1,0,0,0,0,1,0,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0	Trabajador_2
0,0,0,0,0,0,0,0,1,1,0,0,1,0,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0	Trabajador_2
0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0	Trabajador_2
0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,1,0,0,0,0,0,1,0	Trabajador_3
0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,1,0,0,0,0,0,1,0	Trabajador_3
0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,0,0,0,0,1,0	Trabajador_3
0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,1,0,0,0,0,0,1,0	Trabajador_3
0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,1,0,0,0,0,0,1,0	Trabajador_3
0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,1,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0	Trabajador_4
0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,1,0,0,0,1,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0	Trabajador_4
0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,1,0,0,0,1,1,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0	Trabajador_4
0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,0,0,0,1,1,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0	Trabajador_4
0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,0,0,0,1,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0	Trabajador_4
0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,1,0,0,1	Trabajador_5
0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,1,1	Trabajador_5
0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,1,1,0	Trabajador_5
0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,1,0,1	Trabajador_5
0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,1,1,0	Trabajador_5
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0	Trabajador_6
0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0	Trabajador_6
0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,1,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0	Trabajador_6
0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,1,0,1,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0	Trabajador_6
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0	Trabajador_6
0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0	Trabajador_7
0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0	Trabajador_7
0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,1,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0	Trabajador_7
0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,1,0,0,0,1,0,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0	Trabajador_7
0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,1,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	Trabajador_7
1,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	Trabajador_8
1,0,0,0,1,1,0,0,0,0,0,0,0,1,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	Trabajador_8
1,0,0,0,1,0,0,0,0,0,0,0,0,1,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	Trabajador_8
1,0,0,0,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	Trabajador_8
1,1,0,0,1,1,0,0,0,0,0,0,0,1,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	Trabajador_8
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,1,0,0,0,0,0	Trabajador_9
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,0,0,0,0	Trabajador_9
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,0,0,0,0	Trabajador_9
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,1,1,0,0,0,0	Trabajador_9
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,0,0,0,0,0	Trabajador_9
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0	Trabajador_1
0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0	Trabajador_1
0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,0	Trabajador_1
0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,1,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	Trabajador_1
0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,1,0,1	Trabajador_1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0	Trabajador_1
0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	Trabajador_1
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	Trabajador_1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,0,0,0,0,0	Trabajador_1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,0,0,0,0,0	Trabajador_1

Figura 28 - Datasheet y etiquetas

Una vez obtenidos estos datos, se realizan dos pruebas diferentes, la primera se trata de introducirlos en un entorno llamado Weka[15], que se trata de una aplicación que se utiliza para realizar pruebas de diferentes tipos de algoritmos y así poder estudiar su eficacia. Durante estas pruebas (ver figura 29), se selecciona el algoritmo KNN, se introduce el *datasheet*, y se selecciona como parámetro k=3. Se realizan pruebas entre k=1 y k=3 y no se detectan grandes cambios probablemente debido al tamaño del *datasheet*. Se obtiene que el resultado de instancias del test clasificadas es el **70%**.

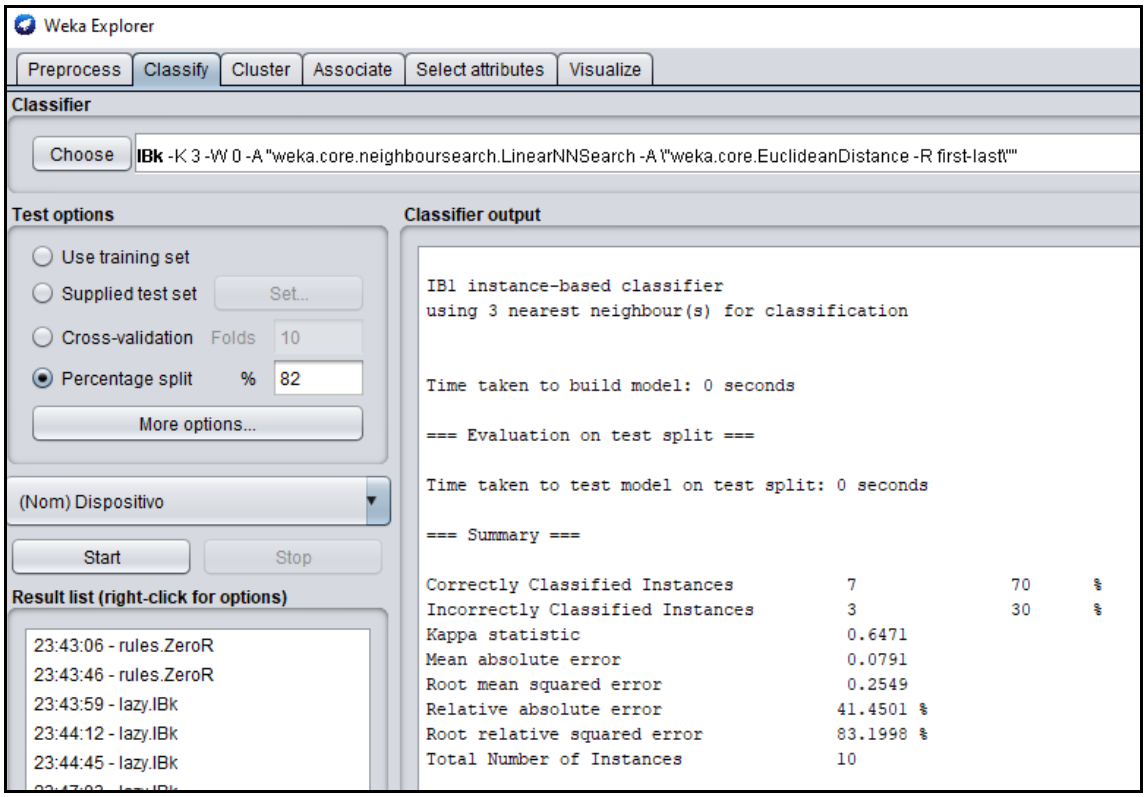


Figura 29 - Resultado del test en Weka

La segunda prueba se realiza directamente en código Python y servirá como preámbulo a la implementación de este algoritmo al código base del programa. Durante esta prueba se desarrolla un script, llamado ai.py (ver figura 30), que se encargará de formatear los datos desde el archivo “vectores.csv” y “etiquetas.csv” y entrenará al sistema con estos datos. Una vez entrenado, se ha creado otro archivo “amenaza.csv” donde se escribe un vector que se

[15] WEKA. https://www.cs.waikato.ac.nz/ml/weka/

inyectará al sistema simulando un ataque para que éste trate de clasificarlo correctamente.

```
Autor: Miguel Ángel Martínez de Castilla de la Mata
Director: Ferran Adelantado i Freixer
Grado: Ingeniería de Tecnologías y Servicios de Telecomunicación
Area: Redes Inalámbricas
TFG: SISTEMA ALEJANDRA (PRUEBAS DE CONCEPTO)
'''

#Este archivo contiene el código del algoritmo de Inteligencia Artificial.
#Importado de librerias
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
import pandas as pd


knn = KNeighborsClassifier(n_neighbors=3) #Se define el número de "vecinos" sobre el que se calcula la distancia

#Lectura de archivos y formateo de datos
a = pd.read_csv('vectores.csv')
b = pd.read_csv('etiquetas.csv')
y = b['Personal'].values
x = a.values


print("Vectores de atributos: ")
print (x)

print("Clases: ")
print(y)

#Definición del metodo de entrenamiento
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.18, random_state=21, stratify=y)


knn.fit(x_train, y_train)#Llamada de entrenamiento

scores = knn.score(x_test,y_test) #Cálculo de precisión

print ("La precisión del algoritmo es: "+str(scores*100)[:5]+"%")

pred1 = pd.read_csv('amenaza.csv')
pred = pred1.values

y_pred = knn.predict(pred) #Se intruduce amenaza para intentar clasificarla
prob = knn.predict_proba(pred) #Se calculan las probabilidades sobre las clases

print ("Vector a evaluar: ")
print(pred)

print ("El vector introducido pertenece al "+str(y_pred[0]))
contador=1
print("Con la siguiente probabilidad:")
for i in prob[0]:
    print ("Trabajador"+str(contador)+": "+ str(i*100)[:5]+"%")
    contador+=1
```

Figura 30 - Código del módulo de ai.py

Como resultado de la ejecución de este sobre el entorno semi artificial creado, se obtiene:

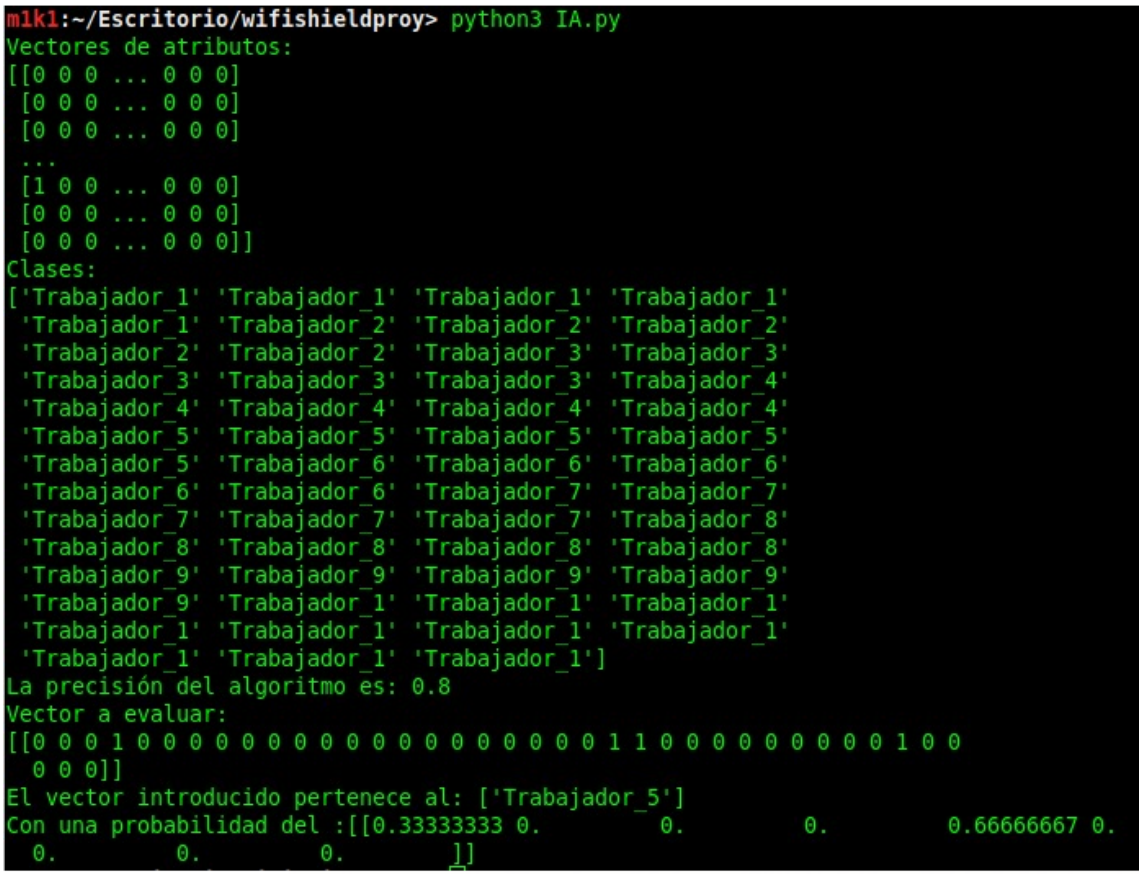


Figura 31 - Resultado de ejecución del módulo ai.py

Se puede apreciar que el código marca una precisión del algoritmo en base a los datos de entrenamiento y test que le hemos introducido del **80%**. Además en lo referente al vector introducido para su clasificación vemos que lo clasifica como del **Trabajador 5**, y posteriormente se ve el vector de probabilidades respecto a la clasificación, en el que nos da un **33% de probabilidades de que sea del trabajador 1 y un 66% de que sea del Trabajador 5.**

5.3 Implementación del algoritmo al Sistema Alejandra

Tras las pruebas de concepto, se procede a realizar una implementación del código que maneja el algoritmo de inteligencia artificial dentro de la versión 1.0 del Sistema Alejandra.

El funcionamiento del Sistema tras la implementación del nuevo módulo será similar al Sistema inicial, con la funcionalidad añadida de que será capaz de discernir si una amenaza es real o puede ser un falso positivo (ver figura 32)

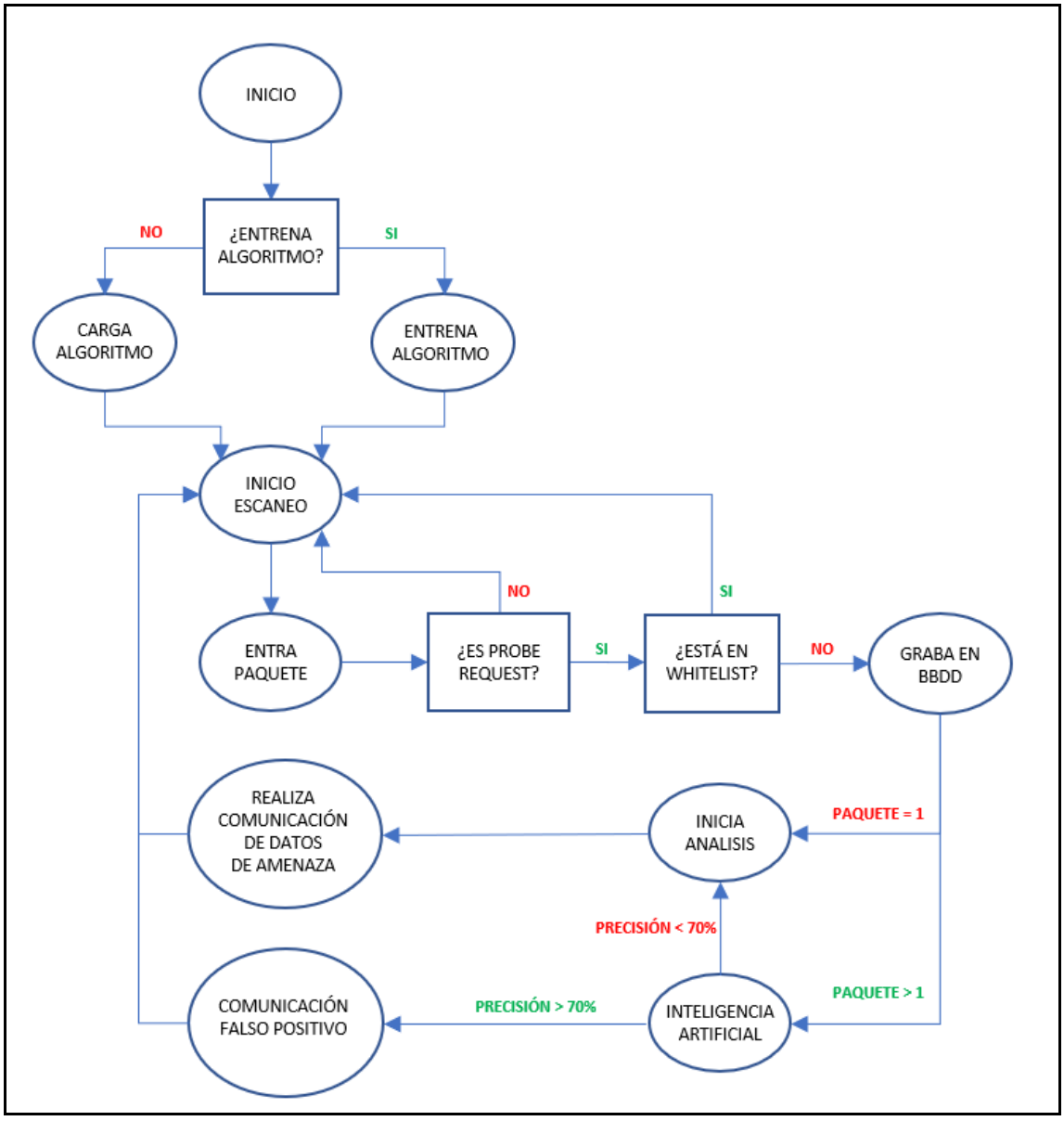


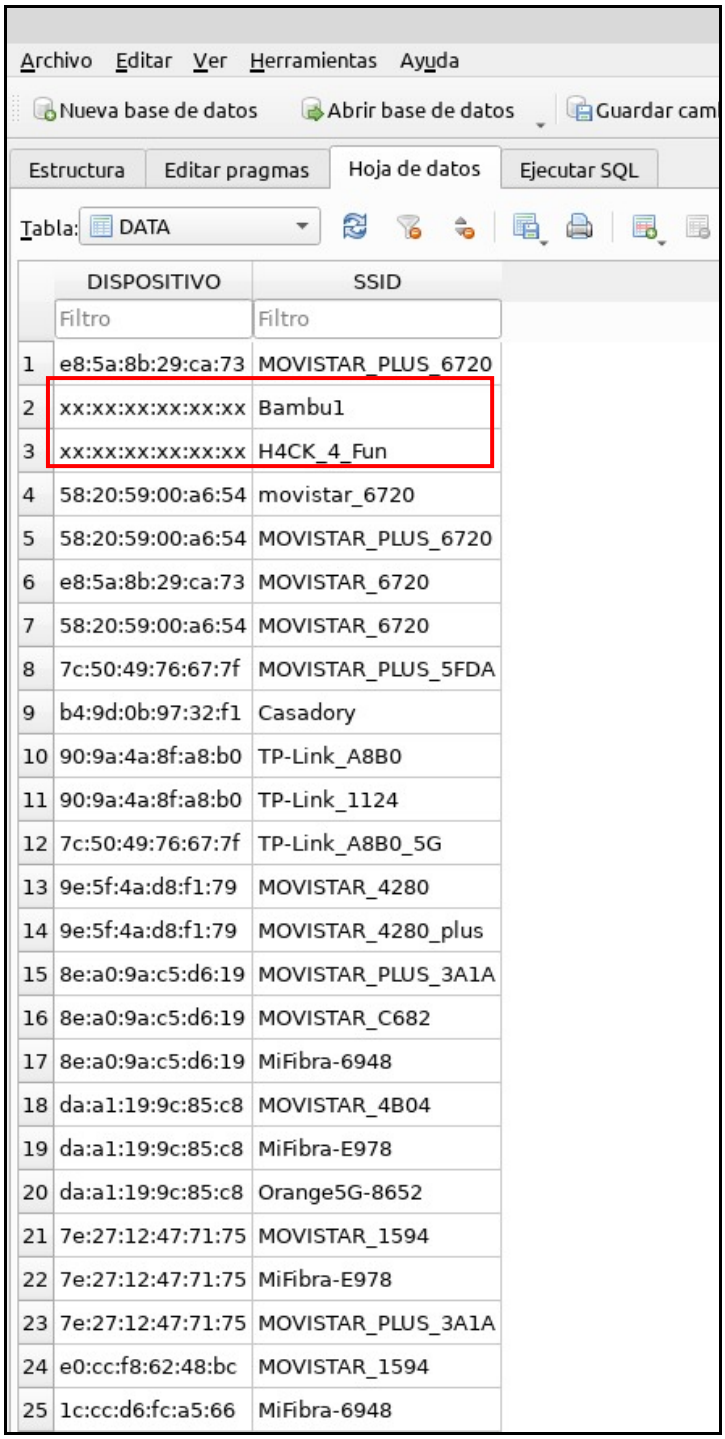
Figura 32 - Diagrama de flujo del Sistema ALEJANDRA v1.1

“*¿En qué caso se pueden dar falsos positivos?”* Esto puede ocasionarse cuando un trabajador de las instalaciones cambie por ejemplo de teléfono móvil y no comunique este hecho al departamento de seguridad. En estas circunstancias, el sistema detectaría un nuevo dispositivo, pero al acceder a redes similares a las vinculadas a los dispositivos pertenecientes al trabajador, la Inteligencia Artificial lo relacionaría con esta persona, pudiendo comprobarse si se trata de la supuesta falta de comunicación del cambio de dispositivo.

Para conseguir esto, en se crea un nuevo módulo de inicio denominado *SmartAlejandra.py* que en vez de cargar o crear una base de datos con solo una *whitelist*, entrenará o cargará el sistema de Inteligencia Artificial que gestionará las amenazas.

En este caso, en una fase previa al arranque, se necesitará tener cargados en la base de datos, *data.db*, los dispositivos autorizados y sus redes conocidas. De aquí se extraerá, por un lado, los dispositivos conocidos almacenándolos en el archivo “*DISPOSITIVOS*”, y por otro las redes conocidas totales existentes almacenándolas en el archivo “*SSIDs*”. Una vez hecho, se utilizará el script “clasificación.py” detallado en el apartado anterior con el que se obtendrá el archivo “*vectores.csv*”, que contendrá una serie de vectores en forma de ceros y unos que serán la base del entrenamiento del algoritmo de *machine learning.*

Para el desarrollo del presente proyecto, como ya se ha mencionado en el apartado anterior, se ha manipulado este archivo debido a la complejidad de obtener datos fiables en un entorno doméstico. Igualmente se realiza un cambio en el mismo sustituyendo el valor de la MAC “b4:8b:19:94:65:53” por “xx:xx:xx:xx:xx:xx” (ver figura 33) para realizar una simulación de cambio de dispositivo de un trabajador.



	DISPOSITIVO	SSID
	Filtro	Filtro
1	e8:5a:8b:29:ca:73	MOVISTAR_PLUS_6720
2	xx:xx:xx:xx:xx:xx	Bambu1
3	xx:xx:xx:xx:xx:xx	H4CK_4_Fun
4	58:20:59:00:a6:54	movistar_6720
5	58:20:59:00:a6:54	MOVISTAR_PLUS_6720
6	e8:5a:8b:29:ca:73	MOVISTAR_6720
7	58:20:59:00:a6:54	MOVISTAR_6720
8	7c:50:49:76:67:7f	MOVISTAR_PLUS_5FDA
9	b4:9d:0b:97:32:f1	Casadory
10	90:9a:4a:8f:a8:b0	TP-Link_A8B0
11	90:9a:4a:8f:a8:b0	TP-Link_1124
12	7c:50:49:76:67:7f	TP-Link_A8B0_5G
13	9e:5f:4a:d8:f1:79	MOVISTAR_4280
14	9e:5f:4a:d8:f1:79	MOVISTAR_4280_plus
15	8e:a0:9a:c5:d6:19	MOVISTAR_PLUS_3A1A
16	8e:a0:9a:c5:d6:19	MOVISTAR_C682
17	8e:a0:9a:c5:d6:19	MiFibra-6948
18	da:a1:19:9c:85:c8	MOVISTAR_4B04
19	da:a1:19:9c:85:c8	MiFibra-E978
20	da:a1:19:9c:85:c8	Orange5G-8652
21	7e:27:12:47:71:75	MOVISTAR_1594
22	7e:27:12:47:71:75	MiFibra-E978
23	7e:27:12:47:71:75	MOVISTAR_PLUS_3A1A
24	e0:cc:f8:62:48:bc	MOVISTAR_1594
25	1c:cc:d6:fc:a5:66	MiFibra-6948

Figura 33 - Extractor de la BBDD "data.db"

Al iniciar el Sistema Alejandra, el programa da opción de entrenar un entorno de Inteligencia Artificial, o cargar uno existente que se haya guardado previamente por el sistema en el archivo “*knn_entrenado*” (ver figura 34).

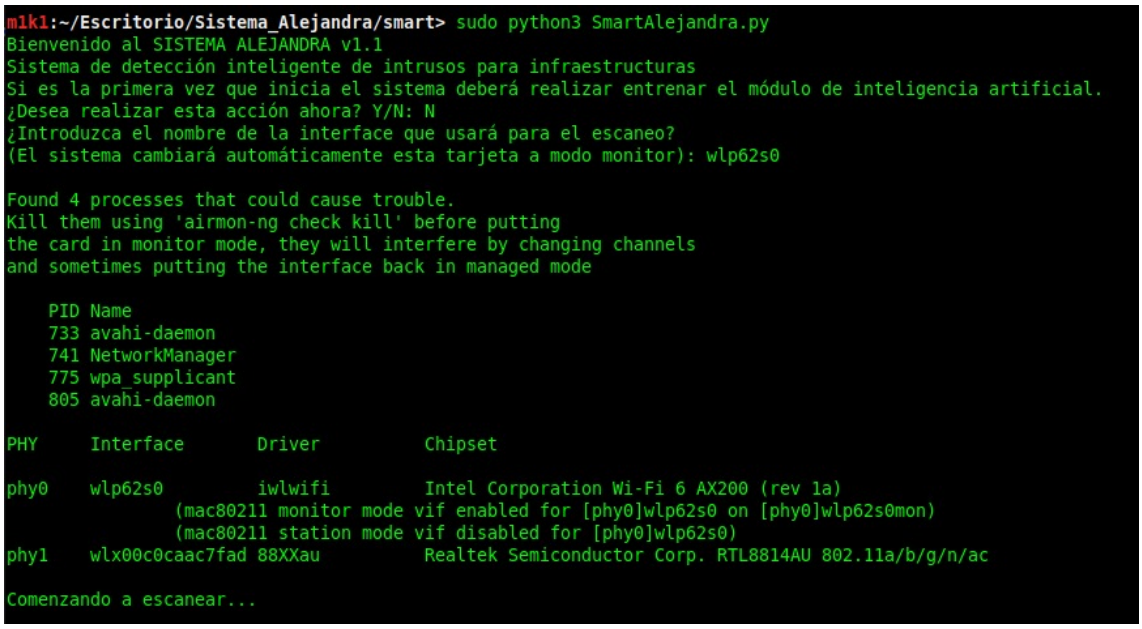


Figura 34 - Inicio Sistema ALEJANDRA v1.1

Tras seleccionar lo deseado, el sistema comienza a escanear. El proceso de descartar amenazas no se detalla más allá ya que es el mismo que el de la versión v.1.0, haciendo peticiones a la base de datos y cruzando información con las MACs autorizadas. En lo referente a las amenazas, el flujo de los datos se controla en tres fases diferentes:

- La primera, es cuando el sistema detecta por primera vez una amenaza (ver figura 35). En este caso el sistema se comporta como la versión 1.0 informando al responsable de seguridad de un dispositivo no autorizado junto con el análisis efectuado del mismo (ver figura 36). Se realiza de esta manera porque en un entorno real no es fiable un único registro para dar fiabilidad al algoritmo de Inteligencia Artificial.

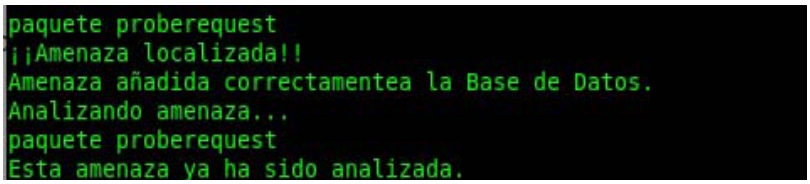
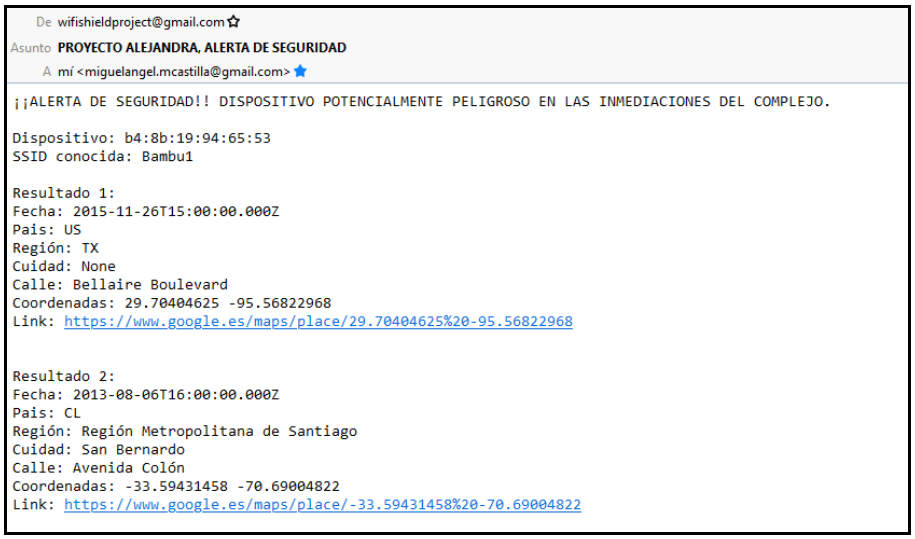


Figura 35 - Extracto del log en pantalla de la localización de amenaza.


De wifishieldproject@gmail.com

Asunto **PROYECTO ALEJANDRA, ALERTA DE SEGURIDAD**

A mí <miguelangel.mcastilla@gmail.com>

¡¡ALERTA DE SEGURIDAD!! DISPOSITIVO POTENCIALMENTE PELIGROSO EN LAS INMEDIACIONES DEL COMPLEJO.

Dispositivo: b4:8b:19:94:65:53
SSID conocida: Bambu1

Resultado 1:
Fecha: 2015-11-26T15:00:00.000Z
Pais: US
Región: TX
Cuidad: None
Calle: Bellaire Boulevard
Coordenadas: 29.70404625 -95.56822968
Link: https://www.google.es/maps/place/29.70404625%20-95.56822968

Resultado 2:
Fecha: 2013-08-06T16:00:00.000Z
Pais: CL
Región: Región Metropolitana de Santiago
Cuidad: San Bernardo
Calle: Avenida Colón
Coordenadas: -33.59431458 -70.69004822
Link: https://www.google.es/maps/place/-33.59431458%20-70.69004822

Figura 36 - Email de comunicación de la primera amenaza.

- La segunda, es cuando se recibe otro SSID conocido de una amenaza. En este caso el sistema activa el módulo de Inteligencia Artificial (ver figura 37), y si este logra una precisión menor al 70%, realiza un análisis igual al anterior ampliando datos mediante un email al responsable de seguridad sobre nuevas ubicaciones obtenidas (ver figura 38).

Figura 37 - Extracto del log en pantalla de la localización de amenaza.


De wifishieldproject@gmail.com

Asunto **PROYECTO ALEJANDRA, ALERTA DE SEGURIDAD(AMPLIANDO INFORMACION)**

A mí <miguelangel.mcastilla@gmail.com>

¡¡ALERTA DE SEGURIDAD!! ALEJANDRA NO HA ENCONTRADO COINCIDENCIAS SUFICIENTES CON NINGUN TRABAJADOR POR LO QUE SE AMPLIAN DATOS DE LA AMENAZA.

Dispositivo: b4:8b:19:94:65:53
SSID conocida: H4CK_4_Fun

Resultado 1:
Fecha: 2017-11-13T13:00:00.000Z
Pais: ES
Región: Comunidad de Madrid
Cuidad: Madrid
Calle: Calle de Aragón
Coordenadas: 40.44960022 -3.58063626
Link: https://www.google.es/maps/place/40.44960022%20-3.58063626

Resultado 2:
Fecha: 2020-08-14T12:00:00.000Z
Pais: ES
Región: Comunidad de Madrid
Cuidad: Madrid
Calle: Avenida de Aragón
Coordenadas: 40.44961929 -3.58088446
Link: https://www.google.es/maps/place/40.44961929%20-3.58088446

Resultado 3:
Fecha: 2016-03-31T14:00:00.000Z
Pais: ES
Región: Comunidad de Madrid
Cuidad: Madrid
Calle: None
Coordenadas: 40.44966125 -3.58130693
Link: https://www.google.es/maps/place/40.44966125%20-3.58130693

Figura 38 - Email de comunicación de la ampliación de datos de la amenaza.

- Por último, si se recibe una nueva red conocida (ver figura 39), y se obtiene una precisión mayor al 70%, el sistema comunica un posible falso positivo al responsable de seguridad (ver figura 40) y a que persona autorizada puede pertenecer el dispositivo, así como la precisión de cálculo. Esto motivaría que el responsable de seguridad se dirija a la persona autorizada para verificar si efectivamente se trata de un falso positivo.

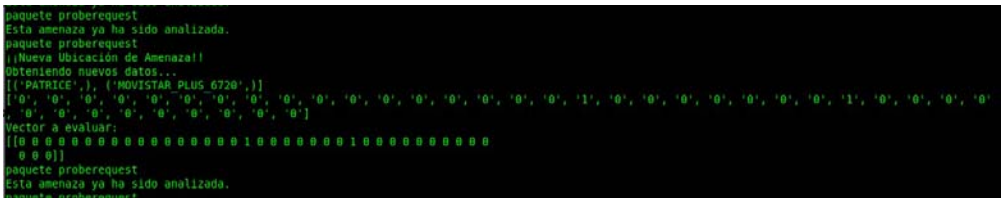


Figura 39 - Extracto del log en pantalla de la localización de amenaza.

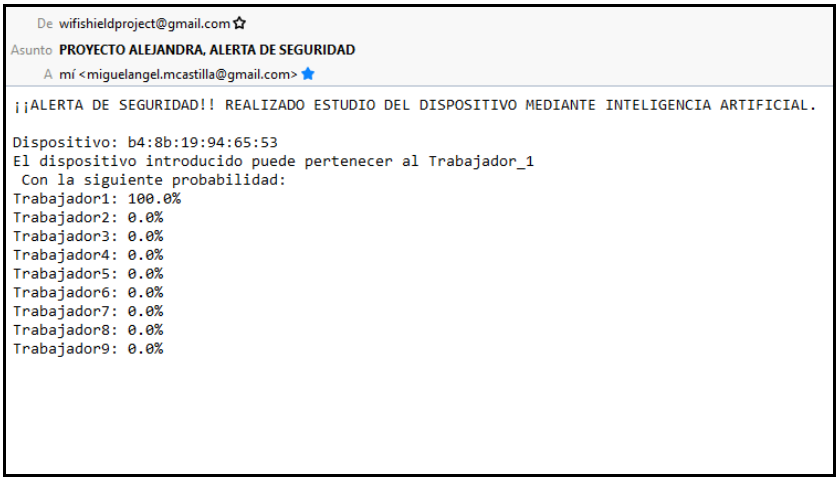


Figura 40 - Email de comunicación de falso positivo.

6. Conclusiones y futuras investigaciones.

Este proyecto nace como una necesidad de ampliación de la seguridad de infraestructuras críticas, y como conclusión tras el desarrollo de este se puede obtener que es un sistema estable, que realiza la función para la que está pensado. En su versión 1.0, el sistema es capaz de detectar amenazas comparando correctamente con la *Whitelist* y una vez localizada, la analiza obteniendo los datos de la manera esperada y comunicándolo vía email.

Posteriormente en su versión 1.1, el sistema realiza correctamente el algoritmo de Inteligencia Artificial clasificando la amenaza como positivo o falso positivo con un rango del 80% de precisión. Esta clasificación se basa en si la probabilidad de que sea un trabajador es mayor o no a un 70%.

Si bien el resultado del sistema es óptimo, se encuentran varios puntos con posibles mejoras que convendrían ser tratadas en futuras líneas de investigación.

- El primero, tratar de desarrollar el programa bajo un procesado de hilos para capturar varios paquetes paralelamente pudiendo evitar perder algunos de interés.

- Desarrollo de un módulo que escanee paquetes de dispositivos BLE.

- Mejoras del algoritmo de Inteligencia Artificial.

- Optimizar la comunicación de amenazas. En la actualidad se realiza una comunicación por paquete identificado como “no autorizado” recibido. Realizar esta optimización mejoraría la experiencia de usuario si se consiguiese acotar esa amenaza mucho más antes de comunicarla. Probablemente el desarrollo utilizando hilos de procesado mejore este punto.

- Por último, aunque mucho más interesante, se podría desarrollar que el sistema levantase un punto de acceso con el nombre de una de las redes conocidas del dispositivo “no autorizado” y permitir que este se

conecte a él automáticamente. De esta manera interactuar directamente con el dispositivo con el fin de obtener la mayor cantidad de datos posibles de este.

*“Cabe reseñar que en un primer momento el proyecto iba a ser llamado “WifiShield”, pero posteriormente se cambió el mismo a “**Sistema ALEJANDRA**”, y es por ello que ciertas capturas de pantalla de ejecuciones de código o recursos utilizados como la dirección email tienen esas referencias”*

7. Bibliografía

- IEEE https://www.ieee.org/ [Fecha de consulta: 20-03-2021]
- Wifi Alliance https://www.wi-fi.org/ [Fecha de consulta: 20-03-2021]
- INE. https://www.ine.es/prensa/tich_2017.pdf [Fecha de consulta: 21-03-2021]
- https://www.netspotapp.com/es/wifi-channel-scanner.html [Fecha de consulta: 01-04-2021]
- https://www.oreilly.com/library/view/80211-wireless-networks/0596100523/ch04.html [Fecha de consulta: 27-03-2021]
- https://www.oreilly.com/library/view/80211ac-a-survival/9781449357702/ch03.html [Fecha de consulta: 01-03-2021]
- GÉRON, Aurélien. Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems. Lugar de publicación: Sebastopol CA Editorial, O`really Media, Inc. 2019. ISBN. [Consulta: 02-04-2021]. Disponible en: https://es1lib.org/book/5207527/b6e3a9
- Inteligencia Artificial vs Aprendizaje Automático vs Deep Learning vs Ciencia de Datos https://www.eadic.com/inteligencia-artificial-vs-aprendizaje-automatico-vs-deep-learning-vs-ciencia-de-datos/ [Fecha de consulta: 05-04-2021]
- SCAPY https://scapy.net/ [Fecha de consulta: 12-04-2021]
- WITTEN, Ian H. & FRANK Eibe (2005). Data Mining. Practical Machine Learning Tools and Techniques. San Francisco. Elsevier.
- WIGLE. https://www.wigle.com [Fecha de consulta: 12-04-2021]
- WEKA. https://www.cs.waikato.ac.nz/ml/weka/ [Fecha de consulta: 09-05-2021]
- AIRCRACK-NG. https://www.aircrack-ng.org/ [Fecha de consulta: 02-04-2021]

8. Anexos

ANEXO I: CÓDIGO BASE DEL SISTEMA ALEJANDRA V. 1.0.

- Alejandra.py:

```
Autor: Miguel Ángel Martínez de Castilla de la Mata
Director: Ferran Adelantado i Freixer
Grado: Ingeniería de Tecnologías y Servicios de Telecomunicación
Area: Redes Inalámbricas
TFG: SISTEMA ALEJANDRA v1.0
'''

#Este archivo contiene el código de funcionamiento principal del sistema.

#Importación de módulos y librerias
import time
import os
import comunicacion
import sniffer
import procesado
import analisis

#Arranque del programa
print ("Bienvenido a WIFISHIELD\nSistema de detección inteligente de intrusos para infraestructuras")
print ("Si es la primera vez que inicia el sistema deberá crear las bases de datos necesarias.")
bases = input("¿Desea crear las bases de datos ahora? Y/N: ")

while ((bases != "Y") and (bases != "y") and (bases != "N") and (bases != "n")):
    bases = input("Opcion incorrecta, ¿Desea crear las bases de datos ahora? Y/N: ")



if ((bases == "Y") or (bases == "y")):
    procesado.crearBases()
    print("Base de datos creada correctamente.")
elif ((bases == "N") or (bases == "n")):
    procesado.conectarBases()
else:
    print("Opción erronea, reinicie el programa y elija una opción correcta.")

interface = input("¿Introduzca el nombre de la interface que usará para el escaneo?\n(El sistema cambiará automáticamente esta tarjeta a modo monitor): ")

os.system("airmon-ng start "+interface)

interface=interface+"mon"
time.sleep(2)
print("Comenzando a escanear...\n")

#Comienza el scaneo.
sniffer.startSniff(interface)
```

Figura 41 - Código del módulo Alejandra.py

- Sniffer.py:

```
Autor: Miguel Ángel Martínez de Castilla de la Mata
Director: Ferran Adelantado i Freixer
Grado: Ingeniería de Tecnologías y Servicios de Telecomunicación
Area: Redes Inalámbricas
TFG: SISTEMA ALEJANDRA v1.0
'''

#Este archivo contiene el código de funcionamiento del monitoreo de paquetes.

#Importación de módulos y librerias
import sys
from scapy.all import *
import procesado
import analisis
import time

#Función que parsea el paquete Probe Request a analizar y hace crossmatch con las Bases de Datos.
def Packet(pkt):

    if pkt.haslayer(Dot11ProbeReq):

        if len(pkt.info) > 0:
            print("paquete proberequest")
            time.sleep(2)
            mac1=str(pkt.addr2)
            ssid=str(pkt.info)[2:len(str(pkt.info))-1]

            if procesado.consultaWhiteList(mac1)==1:
                print("Dispositivo autorizado: "+mac1)

            elif ((procesado.consultaWhiteList(mac1)==0) and (procesado.consultaAmenaza(mac1)==0)):
                procesado.cargaAmenaza(mac1, ssid)
                print ("¡¡Amenaza localizada!!\nAmenaza añadida correctamentea la Base de Datos.\nAnalizando amenaza...")
                time.sleep(1)
                analisis.analizaAmenaza(ssid,mac1)

            elif ((procesado.consultaWhiteList(mac1)==0) and (procesado.consultaAmenaza(mac1)==1) and (procesado.consultaAmenazaInfo(mac1,ssid)==0)):
                procesado.cargaAmenaza(mac1, ssid)
                print ("¡¡Nueva Ubicación de Amenaza!!\nObteniendo nuevos datos...")
                time.sleep(1)
                analisis.analizaAmenaza(ssid,mac1)

            else:
                print("Esta amenaza ya ha sido analizada.")




#Función de scaneo.
def startSniff(interface):

    sniff(iface=interface, count= 0,prn=Packet)
```

Figura 42 – Código del módulo Sniffer.py

- Procesado.py:

```
Autor: Miguel Ángel Martínez de Castilla de la Mata
Director: Ferran Adelantado i Freixer
Grado: Ingeniería de Tecnologías y Servicios de Telecomunicación
Area: Redes Inalámbricas
TFG: SISTEMA ALEJANDRA v1.0
'''

#Este archivo contiene el código de gestión de la base de datos.


#Importación de módulos y librerías
import sqlite3
import time

#Función de creación de BBDD.
def crearBases():

    BaseWifishield= sqlite3.connect('BBDD_Wifishield.db')#Crea la BBDD. Si existe conecta con ella
    cursorWifishield= BaseWifishield.cursor()#Crea el cursor con el que manejaremos la BBDD.
    cursorWifishield.execute("CREATE TABLE WHITELIST (DISPOSITIVO VARCHAR(17))") #Crea la tabla de dispositivos conocidos "WHITELIST".
    cursorWifishield.execute("CREATE TABLE AMENAZAS (DISPOSITIVO VARCHAR(17), SSID VARCHAR(20))") #Crea BB donde se almacenará información de la amenaza.
    cursorWifishield.execute("INSERT INTO WHITELIST VALUES ('2a:a1:43:3f:4e:b8')") #Alimenta la Whitelist
    cursorWifishield.execute("INSERT INTO WHITELIST VALUES ('4e:86:57:3d:b8:26')")
    cursorWifishield.execute("INSERT INTO WHITELIST VALUES ('b0:2a:43:91:47:ca')")
    cursorWifishield.execute("INSERT INTO WHITELIST VALUES ('b2:eb:24:b1:f0:97')")
    cursorWifishield.execute("INSERT INTO WHITELIST VALUES ('00:c0:ca:ac:7f:ad')")
    cursorWifishield.execute("INSERT INTO WHITELIST VALUES ('44:85:00:c9:6d:82')")
    cursorWifishield.execute("INSERT INTO WHITELIST VALUES ('e8:5a:8b:29:ca:73')")
    cursorWifishield.execute("INSERT INTO WHITELIST VALUES ('60:33:4b:65:60:3f')")
    cursorWifishield.execute("INSERT INTO WHITELIST VALUES ('4a:bb:48:6b:87:12')")
    cursorWifishield.execute("INSERT INTO WHITELIST VALUES ('58:20:59:00:a6:54')")
    cursorWifishield.execute("INSERT INTO WHITELIST VALUES ('de:74:db:61:70:d8')")
    cursorWifishield.execute("INSERT INTO WHITELIST VALUES ('94:e6:f7:2b:d7:66')")
    cursorWifishield.execute("INSERT INTO WHITELIST VALUES ('e0:b5:2d:d7:7f:8a')")
    cursorWifishield.execute("INSERT INTO WHITELIST VALUES ('82:5c:bd:6e:0f:84')")
    cursorWifishield.execute("INSERT INTO WHITELIST VALUES ('ee:08:6b:94:65:53')")
    cursorWifishield.execute("INSERT INTO WHITELIST VALUES ('30:3a:64:53:d1:44')")
    cursorWifishield.execute("INSERT INTO WHITELIST VALUES ('32:08:6b:9a:bd:d2')")
    cursorWifishield.execute("INSERT INTO WHITELIST VALUES ('12:15:1b:f4:a7:52')")


    BaseWifishield.commit()
```

Figura 43 - Código del módulo Procesado.py (Parte I)

```
#Función de conexión de BBDD.
def conectarBases():

    BaseWifishield= sqlite3.connect('BBDD_Wifishield.db')#Crea la BBDD. Si existe conecta con ella
    cursorWifishield= BaseWifishield.cursor()#Crea el cursor con el que manejaremos la BBDD.


#Función de consulta a la Whitelist.
def consultaWhiteList(mac):


    BaseWifishield= sqlite3.connect('BBDD_Wifishield.db')
    cursorWifishield= BaseWifishield.cursor()
    cursorWifishield.execute("SELECT DISPOSITIVO FROM WHITELIST")
    lista=cursorWifishield.fetchall()
    match=0
    for a in lista:
        if mac in a:

            match=1

    return match

#Función de creación de nueva línea Amenaza en la BBDD.
def cargaAmenaza(mac,info):
    BaseWifishield= sqlite3.connect('BBDD_Wifishield.db')
    cursorWifishield= BaseWifishield.cursor()
    cursorWifishield.execute("INSERT INTO AMENAZAS (DISPOSITIVO,SSID) VALUES (?,?)",(mac,info))
    BaseWifishield.commit()


#Función de consulta a la BBDD de Amenazas para ver si ese SSID ya existe.
def consultaAmenazaInfo(mac, info):

    BaseWifishield= sqlite3.connect('BBDD_Wifishield.db')
    cursorWifishield= BaseWifishield.cursor()
    cursorWifishield.execute("SELECT SSID FROM AMENAZAS where DISPOSITIVO=?",(mac,))
    wifis=cursorWifishield.fetchall()
    match=0
    for a in wifis:
        if info in a:

            match=1
    return match

#Función de consulta a la BBDD de Amenazas.
def consultaAmenaza(mac):

    BaseWifishield= sqlite3.connect('BBDD_Wifishield.db')
    cursorWifishield= BaseWifishield.cursor()
    cursorWifishield.execute("SELECT DISPOSITIVO FROM AMENAZAS")
    amenaza=cursorWifishield.fetchall()
    match=0
    for a in amenaza:
        if mac in a:

            match=1

    return match
```

Figura 44 - Código de módulo Procesado.py (Parte II)

- Analisis.py:

```
Autor: Miguel Ángel Martínez de Castilla de la Mata
Director: Ferran Adelantado i Freixer
Grado: Ingeniería de Tecnologías y Servicios de Telecomunicación
Area: Redes Inalámbricas
TFG: SISTEMA ALEJANDRA v1.0
'''

#Este archivo contiene el código de funcionamiento de análisis de las redes encontradas.

#Importación de módulos y librerias
import sys
import comunicacion
import requests

#Función que conecta con WIGLE y realiza la presentación de datos para la comunicación.
def analizaAmenaza(ssid,mac):

    url= 'https://api.wigle.net/api/v2/network/search?ssid='
    usuario= 'AID768a3e53108648827e3a2ae1220acc87'
    password= '846f4909cbbd03ba73302c51b1b5c248'

    r=requests.get(url+ssid, auth=(usuario, password))
    resultado=r.json().get('results')
    contador=1
    envio=[]

    for i in resultado:
        a=str(contador)
        lat=str(i.get('trilat'))
        longit=str(i.get('trilong'))
        pais=str(i.get('country'))
        ciudad=str(i.get('city'))
        calle=str(i.get('road'))
        fecha=str(i.get('lastupdt'))
        region=str(i.get('region'))
        envio.append("\n\nResultado "+ a+":"+
            "\nFecha: "+fecha+
            "\nPais: "+pais+
            "\nRegión: "+region+
            "\nCuidad: "+ciudad+
            "\nCalle: "+calle+
            "\nCoordenadas: " +lat+" "+longit+
            "\nLink: https://www.google.es/maps/place/"+lat+"%20"+longit)

        contador += 1

    datos="\n".join(envio)

    comunicacion.mandaemail(datos,mac,ssid)
```

Figura 45 - Código del módulo Analisis.py

- Comunicacion.py:

```
Autor: Miguel Ángel Martínez de Castilla de la Mata
Director: Ferran Adelantado i Freixer
Grado: Ingeniería de Tecnologías y Servicios de Telecomunicación
Area: Redes Inalámbricas
TFG: SISTEMA ALEJANDRA v1.0
'''

#Este archivo contiene el código para realizar la comunicación vía email.

#Importación de módulos y librerias
import smtplib
from email.mime.text import MIMEText

#Función que realiza la comunicación via Email con el responsable de seguridad.
def mandaemail(datos,mac,ssid):
    emisor = "wifishieldproject@gmail.com"
    receptor = "miguelangel.mcastilla@gmail.com"
    amenaza= "Dispositivo: "+mac+"\nSSID conocida: "+ssid

    # Configuracion del mail
    mensaje = MIMEText("¡¡ALERTA DE SEGURIDAD!! DISPOSITIVO POTENCIALMENTE PELIGROSO EN LAS INMEDIACIONES DEL COMPLEJO.\n\n"+amenaza+datos)
    mensaje['From']=emisor
    mensaje['To']=receptor
    mensaje['Subject']="WIFISHIELD, ALERTA DE SEGURIDAD"

    # Nos conectamos al servidor SMTP de Gmail
    serverSMTP = smtplib.SMTP('smtp.gmail.com',587)
    serverSMTP.ehlo()
    serverSMTP.starttls()
    serverSMTP.ehlo()
    serverSMTP.login(emisor,".a123456")

    # Enviamos el mensaje
    serverSMTP.sendmail(emisor,receptor,mensaje.as_string())

    # Cerramos la conexion
    serverSMTP.close()
```

Figura 46 - Código del módulo Comunicacion.py

ANEXO II: CODIGO SISTEMA ALEJANDRA V. 1.1.

- SmartAlejandra.py:

```
'''
Autor: Miguel Ángel Martínez de Castilla de la Mata
Director: Ferran Adelantado i Freixer
Grado: Ingeniería de Tecnologías y Servicios de Telecomunicación
Area: Redes Inalámbricas
TFG: SISTEMA ALEJANDRA v1.1
'''

#Este archivo contiene el código de funcionamiento principal del sistema.

#Importación de módulos y librerias
import time
import os
import comunicacion
import sniffer
import procesado
import analisis
import ai

#Arranque del programa
print ("Bienvenido al Proyecto ALEJANDRA\nSistema de detección inteligente de intrusos para infraestructuras")
print ("Si es la primera vez que inicia el sistema deberá realizar entrenar el módulo de inteligencia artificial.")
bases = input(";¿Desea realizar esta acción ahora? Y/N: ")

while ((bases != "Y") and (bases != "y") and (bases != "N") and (bases != "n")):
    bases = input("Opcion incorrecta, ¿Desea realizar el entrenamiento del sistema ahora? Y/N: ")
    print (bases)


if ((bases == "Y") or (bases == "y")):
    ai.entrenarAlgoritmo()

elif ((bases == "N") or (bases == "n")):
    procesado.conectarBases()

interface = input("¿Introduzca el nombre de la interface que usará para el escaneo?\n(El sistema cambiará automáticamente esta tarjeta a modo monitor): ")

os.system("airmon-ng start "+interface)

interface=interface+"mon"
time.sleep(2)
print("Comenzando a escanear...\n")

#Comienza el scaneo.
sniffer.startSniff(interface)
```

Figura 47 - Código del módulo SmartAlejandra.py

- Ai.py:

```
Autor: Miguel Ángel Martínez de Castilla de la Mata
Director: Ferran Adelantado i Freixer
Grado: Ingeniería de Tecnologías y Servicios de Telecomunicación
Area: Redes Inalámbricas
TFG: SISTEMA ALEJANDRA v1.1
'''

#Este archivo contiene el código del algoritmo de Inteligencia Artificial.
#Importado de librerias
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
import pandas as pd
import pickle
import comunicacion
import analisis

#Funcion para entrenar el algoritmo
def entrenarAlgoritmo():

    knn = KNeighborsClassifier(n_neighbors=3) #Se define el número de "vecinos" sobre el que se calcula la distancia
    a = pd.read_csv('vectores.csv')
    b = pd.read_csv('etiquetas.csv')
    y = b['Personal'].values
    x = a.values

    #Definición del metodo de entrenamiento y grabado en un archivo.
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.18, random_state=21, stratify=y)
    knn.fit(x_train, y_train)#Llamada de entrenamiento
    scores = knn.score(x_train, y_train)#Cálculo de precisión
    print ("La precisión del algoritmo es: "+str(scores*100)[:5]+"%")
    knn_salvado = open('knn_entrenado', 'wb')
    pickle.dump(knn, knn_salvado)#Guarda el algoritmo knn en el archivo knn_entrenado
    print("Algoritmo salvado correctamente.")
```

Figura 48 - Código del módulo ai.py (Parte I)

```
#Funcion para evaluar una amenaza
def predecir(mac,ssid):

    knn = pickle.load(open('knn_entrenado', 'rb'))#Carga el algoritmo knn

    pred1 = pd.read_csv('amenaza.csv')
    pred = pred1.values
    y_pred = knn.predict(pred) #Se intruduce amenaza para intentar clasificarla
    prob = knn.predict_proba(pred) #Se calculan las probabilidades sobre las clases

    print ("Vector a evaluar: ")
    print(pred)
    enviar=[]
    enviar.append("El dispositivo introducido puede pertenecer al "+str(y_pred[0])+"\n Con la siguiente probabilidad:\n")
    contador=1
    for i in prob[0]:
        enviar.append("Trabajador"+str(contador)+": "+ str(i*100)[:5]+"%\n")
        contador+=1

    dato="".join(enviar)

    probabilidad=max(prob[0])

    if probabilidad<0.70:

        analisis.analizaAmenazaAI(ssid,mac)

    else:

        comunicacion.mandaEmailInterno(dato,mac,ssid)
```

Figura 49 - Código del módulo ai.py (Parte II)

- Procesado.py:

```
Autor: Miguel Ángel Martínez de Castilla de la Mata
Director: Ferran Adelantado i Freixer
Grado: Ingeniería de Tecnologías y Servicios de Telecomunicación
Area: Redes Inalámbricas
TFG: SISTEMA ALEJANDRA v1.1
'''

#Este archivo contiene el código de gestión de la base de datos.


#Importación de módulos y librerías
import sqlite3
import time
import ai


#Función de conexión de BBDD.
def conectarBases():

    BaseWifishield= sqlite3.connect('data.db')#Crea la BBDD. Si existe conecta con ella
    cursorWifishield= BaseWifishield.cursor()#Crea el cursor con el que manejaremos la BBDD.


#Función de consulta a la Whitelist.
def consultaWhiteList(mac):


    BaseWifishield= sqlite3.connect('data.db')
    cursorWifishield= BaseWifishield.cursor()
    cursorWifishield.execute("SELECT DISPOSITIVO FROM DATA")
    lista=cursorWifishield.fetchall()
    match=0
    for a in lista:
        if mac in a:

            match=1

    return match
```

Figura 50 - Código del módulo Procesado.py (Parte I)

```
#Función de creación de nueva linea Amenaza en la BBDD.
def cargaAmenaza(mac,info):
    BaseWifishield= sqlite3.connect('data.db')
    cursorWifishield= BaseWifishield.cursor()
    cursorWifishield.execute("INSERT INTO AMENAZAS (DISPOSITIVO,SSID) VALUES (?,?)",(mac,info))
    BaseWifishield.commit()


#Función de consulta a la BBDD de Amenazas para ver si ese SSID ya existe.
def consultaAmenazaInfo(mac, info):

    BaseWifishield= sqlite3.connect('data.db')
    cursorWifishield= BaseWifishield.cursor()
    cursorWifishield.execute("SELECT SSID FROM AMENAZAS where DISPOSITIVO=?",(mac,))
    wifis=cursorWifishield.fetchall()
    match=0
    for a in wifis:
        if info in a:

            match=1
    return match

#Función de consulta a la BBDD de Amenazas.
def consultaAmenaza(mac):

    BaseWifishield= sqlite3.connect('data.db')
    cursorWifishield= BaseWifishield.cursor()
    cursorWifishield.execute("SELECT DISPOSITIVO FROM AMENAZAS")
    amenaza=cursorWifishield.fetchall()
    match=0
    for a in amenaza:
        if mac in a:

            match=1

    return match
```

Figura 51 - Código del módulo Procesado.py (Parte II)

```python
def crearVector(mac,ssid):
    amenaza=open("amenaza.csv", 'w')
    datos= sqlite3.connect('data.db')#Crea la BBDD. Si existe conecta con ella
    cursor= datos.cursor()#Crea el cursor con el que manejaremos la BBDD.
    ssids=open('SSIDs', 'r')
    a=[]
    for i in ssids:
        a.append(i.replace("\n",""))


    b=["0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0"]
    c=cursor.execute("SELECT SSID FROM AMENAZAS where DISPOSITIVO=?",(mac,)).fetchall()

    print (c)
    for i in c:
        x=i[0]
        if x in a:
            indice=a.index(x)
            b[indice]="1"
    print(b)
    amenaza.write(",".join(a)+"\n"+",".join(b)+"\n")

    amenaza.close()
    ai.predecir(mac,ssid)
```

Figura 52 - Código del módulo Procesado.py (Parte III)

- Analisis.py:

```
Autor: Miguel Ángel Martínez de Castilla de la Mata
Director: Ferran Adelantado i Freixer
Grado: Ingeniería de Tecnologías y Servicios de Telecomunicación
Area: Redes Inalámbricas
TFG: SISTEMA ALEJANDRA v1.1
'''

#Este archivo contiene el código de funcionamiento de análisis de las redes encontradas.

#Importación de módulos y librerias
import sys
import comunicacion
import requests

#Función que conecta con WIGLE y realiza la presentación de datos para la comunicación.
def analizaAmenaza(ssid,mac):

    url= 'https://api.wigle.net/api/v2/network/search?ssid='
    usuario= 'AID768a3e53108648827e3a2ae1220acc87'
    password= '846f4909cbbd03ba73302c51b1b5c248'

    r=requests.get(url+ssid, auth=(usuario, password))
    resultado=r.json().get('results')
    contador=1
    envio=[]

    for i in resultado:
        a=str(contador)
        lat=str(i.get('trilat'))
        longit=str(i.get('trilong'))
        pais=str(i.get('country'))
        ciudad=str(i.get('city'))
        calle=str(i.get('road'))
        fecha=str(i.get('lastupdt'))
        region=str(i.get('region'))
        envio.append("\n\nResultado "+ a+":"+
            "\nFecha: "+fecha+
            "\nPais: "+pais+
            "\nRegión: "+region+
            "\nCuidad: "+ciudad+
            "\nCalle: "+calle+
            "\nCoordenadas: " +lat+" "+longit+
            "\nLink: https://www.google.es/maps/place/"+lat+"%20"+longit)

        contador += 1

    datos="\n".join(envio)

    comunicacion.mandaEmail(datos,mac,ssid)
```

Figura 53 - Código del módulo Analisis.py (Parte I)

```
#Función que conecta con WIGLE tras realizar comprobación de inteligencia artificial.
def analizaAmenazaAI(ssid,mac):

    url= 'https://api.wigle.net/api/v2/network/search?ssid='
    usuario= 'AID768a3e53108648827e3a2ae1220acc87'
    password= '846f4909cbbd03ba73302c51b1b5c248'

    r=requests.get(url+ssid, auth=(usuario, password))
    resultado=r.json().get('results')
    contador=1
    envio=[]

    for i in resultado:
        a=str(contador)
        lat=str(i.get('trilat'))
        longit=str(i.get('trilong'))
        pais=str(i.get('country'))
        ciudad=str(i.get('city'))
        calle=str(i.get('road'))
        fecha=str(i.get('lastupdt'))
        region=str(i.get('region'))
        envio.append("\n\nResultado "+ a+":"+
            "\nFecha: "+fecha+
            "\nPais: "+pais+
            "\nRegión: "+region+
            "\nCuidad: "+ciudad+
            "\nCalle: "+calle+
            "\nCoordenadas: " +lat+" "+longit+
            "\nLink: https://www.google.es/maps/place/"+lat+"%20"+longit)

        contador += 1

    datos="\n".join(envio)

    comunicacion.mandaAmpliacionEmail(datos,mac,ssid)
```

Figura 54 - Código del módulo Analisis.py (Parte II)

- Sniffer.py:

```
Autor: Miguel Ángel Martínez de Castilla de la Mata
Director: Ferran Adelantado i Freixer
Grado: Ingeniería de Tecnologías y Servicios de Telecomunicación
Area: Redes Inalámbricas
TFG: SISTEMA ALEJANDRA v1.1
'''

#Este archivo contiene el código de funcionamiento del monitoreo de paquetes.

#Importación de módulos y librerias
import sys
from scapy.all import *
import procesado
import analisis
import time

#Función que parsea el paquete Probe Request a analizar y hace crossmatch con las Bases de Datos.
def Packet(pkt):

    if pkt.haslayer(Dot11ProbeReq):

        if len(pkt.info) > 0:
            print("paquete proberequest")
            time.sleep(2)
            mac1=str(pkt.addr2)
            ssid=str(pkt.info)[2:len(str(pkt.info))-1]

            if procesado.consultaWhiteList(mac1)==1:
                print("Dispositivo autorizado: "+mac1)

            elif ((procesado.consultaWhiteList(mac1)==0) and (procesado.consultaAmenaza(mac1)==0)):
                procesado.cargaAmenaza(mac1, ssid)
                print ("¡¡Amenaza localizada!!\nAmenaza añadida correctamentea la Base de Datos.\nAnalizando amenaza...")
                time.sleep(1)
                analisis.analizaAmenaza(ssid,mac1)

            elif ((procesado.consultaWhiteList(mac1)==0) and (procesado.consultaAmenaza(mac1)==1) and (procesado.consultaAmenazaInfo(mac1,ssid)==0)):
                procesado.cargaAmenaza(mac1, ssid)
                print ("¡¡Nueva Ubicación de Amenaza!!\nObteniendo nuevos datos...")
                time.sleep(1)
                procesado.crearVector(mac1,ssid)
                time.sleep(5)


            else:
                print("Esta amenaza ya ha sido analizada.")




#Función de scaneo.
def startSniff(interface):

    sniff(iface=interface, count= 0,prn=Packet)
```

Figura 55 - Código del módulo Sniffer.py

- Comunicacion.py:

```
Autor: Miguel Ángel Martínez de Castilla de la Mata
Director: Ferran Adelantado i Freixer
Grado: Ingeniería de Tecnologías y Servicios de Telecomunicación
Area: Redes Inalámbricas
TFG: SISTEMA ALEJANDRA v1.1
'''

#Este archivo contiene el código para realizar la comunicación vía email.

#Importación de módulos y librerias
import smtplib
from email.mime.text import MIMEText

#Función que realiza la comunicación via Email con el responsable de seguridad cuando aparezca una amenaza.
def mandaEmail(datos,mac,ssid):
    emisor = "wifishieldproject@gmail.com"
    receptor = "miguelangel.mcastilla@gmail.com"
    amenaza= "Dispositivo: "+mac+"\nSSID conocida: "+ssid

    # Configuracion del mail
    mensaje = MIMEText("¡¡ALERTA DE SEGURIDAD!! DISPOSITIVO POTENCIALMENTE PELIGROSO EN LAS INMEDIACIONES DEL COMPLEJO.\n\n"+amenaza+datos)
    mensaje['From']=emisor
    mensaje['To']=receptor
    mensaje['Subject']="PROYECTO ALEJANDRA, ALERTA DE SEGURIDAD"

    # Nos conectamos al servidor SMTP de Gmail
    serverSMTP = smtplib.SMTP('smtp.gmail.com',587)
    serverSMTP.ehlo()
    serverSMTP.starttls()
    serverSMTP.ehlo()
    serverSMTP.login(emisor,".a123456")

    # Enviamos el mensaje
    serverSMTP.sendmail(emisor,receptor,mensaje.as_string())

    # Cerramos la conexion
    serverSMTP.close()
```

Figura 56 - Código del módulo Comunicacion.py (Parte I)

```
#Función que realiza la comunicación via tmail con el responsable de seguridad tras reconocer el dispositivo mediante inteligencia artificial.
def mandaEmailInterno(datos,mac,ssid):
    emisor = "wifishieldproject@gmail.com"
    receptor = "miguelangel.mcastilla@gmail.com"
    amenaza= "Dispositivo: "+mac+"\n"

    # Configuracion del mail
    mensaje = MIMEText("¡¡ALERTA DE SEGURIDAD!! REALIZADO ESTUDIO DEL DISPOSITIVO MEDIANTE INTELIGENCIA ARTIFICIAL.\n\n"+amenaza+datos)
    mensaje['From']=emisor
    mensaje['To']=receptor
    mensaje['Subject']="PROYECTO ALEJANDRA, ALERTA DE SEGURIDAD"

    # Nos conectamos al servidor SMTP de Gmail
    serverSMTP = smtplib.SMTP('smtp.gmail.com',587)
    serverSMTP.ehlo()
    serverSMTP.starttls()
    serverSMTP.ehlo()
    serverSMTP.login(emisor,".a123456")

    # Enviamos el mensaje
    serverSMTP.sendmail(emisor,receptor,mensaje.as_string())

    # Cerramos la conexion
    serverSMTP.close()

#Función que realiza la comunicación via tmail con el responsable de seguridad cuando el resultado de la AI sea negativo y amplíe datos de ubicación sobre la amenaza.
def mandaAmpliacionEmail(datos,mac,ssid):
    emisor = "wifishieldproject@gmail.com"
    receptor = "miguelangel.mcastilla@gmail.com"
    amenaza= "Dispositivo: "+mac+"\nSSID conocida: "+ssid

    # Configuracion del mail
    mensaje = MIMEText("¡¡ALERTA DE SEGURIDAD!! ALEJANDRA NO HA ENCONTRADO COINCIDENCIAS SUFICIENTES CON NINGUN TRABAJADOR POR LO QUE SE AMPLIAN DATOS DE LA AMENAZA.\n\n"+amenaza+datos)
    mensaje['From']=emisor
    mensaje['To']=receptor
    mensaje['Subject']="PROYECTO ALEJANDRA, ALERTA DE SEGURIDAD(AMPLIANDO INFORMACION)"

    # Nos conectamos al servidor SMTP de Gmail
    serverSMTP = smtplib.SMTP('smtp.gmail.com',587)
    serverSMTP.ehlo()
    serverSMTP.starttls()
    serverSMTP.ehlo()
    serverSMTP.login(emisor,".a123456")

    # Enviamos el mensaje
    serverSMTP.sendmail(emisor,receptor,mensaje.as_string())

    # Cerramos la conexion
    serverSMTP.close()
```

Figura 57 - Código del módulo Comunicación.py (Parte II)